LE VIDEOGUIDE PER PROGRAMMARE BENE
5 WEBCAST UFFICIALI MICROSOFT
GRATIS NEL CD "I CORSI DI FORMAZIONE" DA SEGUIRE SUL PC

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90
VERSIONE STANDARD RIVISTA+CD €6,90

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • APRILE 2006 • ANNO X, N.4 (101)

PILOTARE LA WEBCAM

Non è mai stato così semplice grazie a Visual Basic.NET

STRUMENTI Le librerie da importare nei tuoi progetti

REFERENCE La guida all'uso di metodi e proprietà per catturare il flusso delle immagini

PRATICA Un'applicazione completa per acquisire un video e salvarlo sul disco

APPROFONDIMENTO INTERCETTA UN FASCIO DI LUCE CON LA TELECAMERA E REALIZZA IN C# UN GIOCO DI TIRO AL BERSAGLIO

SPECIALE CELLULARI

- **INVIARE SMS A TEMPO** LI SCRIVI PRIMA, INDICHI LA DATA, E IL TELEFONINO LI INVIA PER TE. MAI PIÙ COMPLEANNI DIMENTICATI!
- **NEWS DA PASSEGGIO** ECCO COME PRELEVARE GLI RSS DA UN SITO WEB E VEDERLI QUANDO VUOI TU SUL TELEFONO

.NET ADVANCED

CONTROLLI PERSONALIZZATI

Le tecniche per estendere Visual Studio con moduli fatti su misura per te

C++

USARE LE WXWIDGETS

Le librerie grafiche per programmare con facilità

GRAFICA

SKELETAL ANIMATION

Le tecniche per realizzare animazioni fluide e realistiche

CASI DI STUDIO: BUSINESS

LUNGHE ATTESE ADDIO!

Connessione con la banca assente? Ecco il servizio da utilizzare per mettere le transazioni in coda e gestirle quando possibile

CASI DI STUDIO: ASP.NET

GRAFICA DINAMICA

Crea pulsanti ed altri elementi "On Demand" prendendo le informazioni da un database

VISUAL BASIC 6.0

SQL SERVER 2005 E LE STORED PROCEDURE

Ecco come integrare i tuoi programmi con il nuovo DB di Microsoft

C++

ACE: IL MULTIPIATTAFORMA PER IL NETWORKING

Sviluppa applicazioni per la rete che funzionano su tutti i sistemi operativi

IOPROGRAMMO BY EXAMPLE

Gli esempi guidati per imparare un linguaggio in modo pratico e divertente

C#, Visual Basic.NET
- Come fare drag and drop da una datagridview ad un'altra
- Come posso aggiungere un'immagine agli elementi di una combobox?

Java
- Come posso realizzare applicazioni multilingue?

e molti altri ancora all'interno...

ALGORITMI DI ORDINAMENTO come funzionano e qual è il più adatto per risolvere i tuoi problemi

Anno X - N.ro 4 (101) - Aprile 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, L. Buono, R. Brunetti, F. Grimaldi, F. Lippo, A. Lacava, A. Panella, R. Sidoti, F. Smelzo.

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Elio Monaco

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 • ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/04/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Marzo 2006

Audio Video Foto Bild, A-Team, Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, Distretto di polizia, DVD Magazine, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web, I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Grandi Guide di ioProgrammo, Linux Magazine, Magnum PI, Miami Vice in DVD, MPC, Nightmare, Office Magazine, Play Generation, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection.

▼ Video e Formazione

Mi è capitato, recentemente, di leggere su un noto forum dedicato alla programmazione un thread in cui in modo molto ironico si ripercorreva l'evoluzione dei tool di sviluppo. Nell'ambito di questo thread, veniva citato ioProgrammo come indispensabile compagno di viaggio per "formarsi" all'uso dei diversi strumenti che si sono alternati sul mercato. Ed in effetti il nostro scopo è sempre stato quello di voler fare in una certa misura "formazione". Non possiamo che essere contenti di affermare che anche grazie al nostro contributo è nata una nuova classe di programmatori che, nel corso di quasi 10 anni, ha sostenuto il mercato dello sviluppo in Italia. Non vogliamo con questo dire che quanto c'è di buono nel settore dello sviluppo sia interamente merito nostro, ma sicuramente possiamo gloriarci di aver dato e di voler continuare a dare il nostro contributo alla formazione. È in questo ambito che nasce la collaborazione con Microsoft che da tempo è impegnata nel diffondere le corrette tecniche di programmazione legate ai propri strumenti. Gli MSDN Webcast, sono le registrazioni fedeli degli eventi di formazione tenuti dagli esperti di Microsoft online . Sono tipicamente composti dalla riproduzione delle slide mostrate durante l'evento, accompagnate dal commento del relatore, oltre le slide ci sono varie sezione di demo che mostrano come utilizzare le varie teniche descritte. ioProgrammo ha voluto aggiungere ai propri contenuti quelli prodotti direttamente da MS. Lo scopo è quello di poter ancora contribuire a fare formazione mettendo nelle mani dei lettori uno strumento da consultare rapidamente dai nostri CD. Questo rappresenta per noi l'ennesimo passo verso una nuova frontiera, ovvero quella di voler rendere un po' più multimediale ioProgrammo che adesso può contare su un'informazione cartacea approfondita e facilmente consultabile, un centro di riferimento su internet con il proprio forum *http://forum.ioprogrammo.it* per scambiare informazioni oppure porre domande, e adesso anche un contenuto multimediale importante che come avrete occasione voi stessi di notare rappresenta uno strumento veramente comodo per apprendere le nuove tecniche.

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

OCCHIO ALLA WEBCAM

Crea in Visual Basic un'applicazione che gestisce il video

- ✔ **teoria: le librerie da utilizzare per interfacciare VB con la telecamera**
- ✔ **tecnica: come utilizzare Visual Studio per includere le dll giuste**
- ✔ **pratica: un esempio completo da seguire passo passo per iniziare subito**

pag. 24

MICROSOFT MESSAGE QUEUE

MSMQ è il servizio Windows che facilita la comunicazione asincrona fra applicazioni per tutti gli scenari in cui le parti possono essere irrangiungibili offline o lavorare a velocità diverse **pag. 78**

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

RIVISTA + CD-ROM in edicola

100% SOFTWARE COMPLETO

fra cui...

- C++ DEV C++ 4.9.9.2
- PHP DEV PHP 2.0.13
- PHP 5.1.2
- PHPMYADMIN 2.8.0
- DELPHI LAZARUS 0.9.10
- C# SHARPDEVELOP 2.2.0
- JAVA J2SE 1.5.0 UPDATE 6
- ECLIPSE 3.1.2
- TOOLS MRTG 2.13
- MYSQL 5.0.18
- WXWIDGETS 2.6.2

Prodotti del mese

Ace

La libreria multipiattaforma per la rete

Di Ace ce ne parla in modo approfondito Alfredo Marroccelli nel bell'articolo contenuto in questo stesso numero di ioProgrammo. Si tratta di una libreria che fornisce al C++ uno strato d'astrazione verso il TCP/IP. In parole povere consente di programmare applicazioni che fanno un uso estensivo delle risorse di rete, costruendo uno strato superiore che fa da interfaccia verso il sistema operativo. Questo garantisce la massima portabilità da un sistema all'altro. Di fatto in fase di programmazione sarà semplicemente necessario richiamare le funzioni e i metodi esposti da ACE, sarà poi compito della libreria interfacciare la parte programmativa con il sistema operativo sottostante. Si rivela un utile strumento per il multipiattaforma

[pag.106]

Jasper Reports 1.2.0

Una libreria scritta in java per creare report personalizzati

Una libreria scritta in java per creare report personalizzati. Una libreria per generare report in grado di inviare il proprio output allo schermo, alla stampante o addirittura a file in formato PDF, HTML, XLS, CSV e e XML. Interamente scritta in Java, può facilmente essere utilizzata in un infinità di applicazioni, per generare contenuti dinamici. Ne abbiamo parlato approfonditamente in quache numero precedente di ioProgrammo, ma per l'interesse che il progetto riveste non mancheremo di riparlarne ancora con articoli che ne illustrano in dettaglio tutte le caratteristiche.

[pag.106]

OpenCV

La libreria multipiattaforma per la WebCam

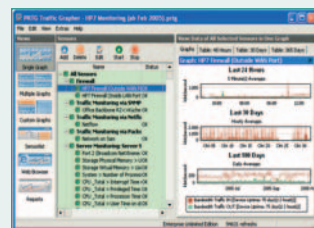
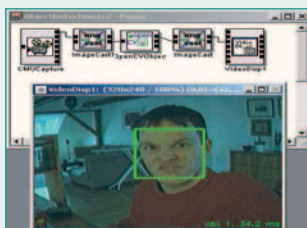
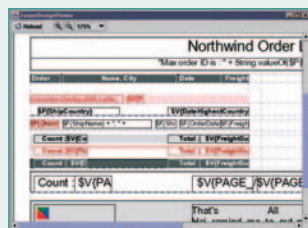
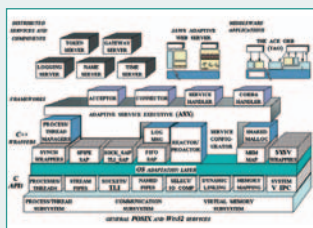
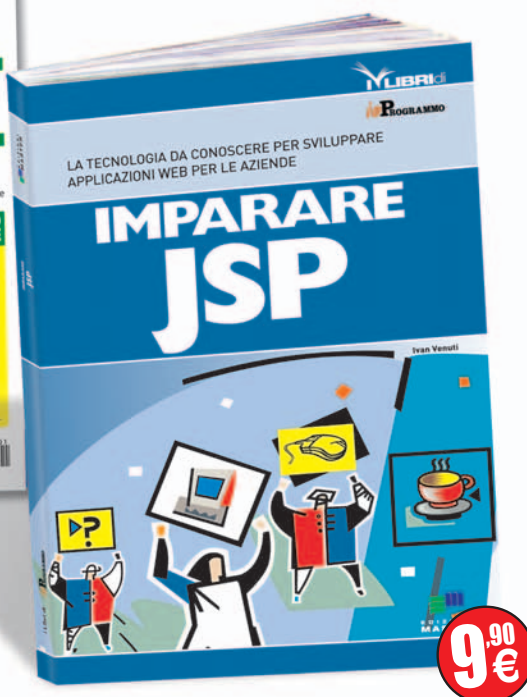
Ce ne parla a lungo Antonino Panella, nell'articolo che questo mese illustra come creare un programma di tiro al bersaglio. L'esempio è didattico, ma le OpenCV sono delle librerie opensource che consentono di gestire in modo ottimale la webcam. Le applicazioni sono infinite, si va dal riconoscimento facciale al motion detection e persino ad usi avanzati e futuristici come il riconoscimento del labiale. Si tratta di librerie estremamente potenti che consentono di lavorare con le immagini in modo sofisticato. Un primo esempio d'uso lo trovate in questo numero, ma siamo convinti che voi stessi sarete in grado di suggerirci impieghi molto fantasioso di questo ottimo framework

[pag.107]

Prtg Traffic Grapher 5.2.0.565

Traffico sotto controllo

Mrtg lo conoscete tutti, si tratta di un software estremamente preciso per determinare il consumo di banda, di processore, di memoria di un sistema. MRTG interroga una macchina usando il protocollo SNMP e questo garantisce una resa ottimale . MRTG ha degli svantaggi. Si tratta di un'applicazione perl che non ha la stessa facilità di utilizzo di un'applicazione a finestre. Per gli utenti Windows c'è PRTG, che tramite una comoda GUI esporta tutte le funzionalità classiche di MRTG in modo comodo e intuitivo, con tanto di grafici e report dettagliati. Molto utile. Senza dubbio un valido aiuto per chi deve tenere sotto controllo una rete, qualunque sia la sua dimensione

[pag.107]

I contenuti del libro

Imparare JSP

Le JSP, ovvero le Java Server Pages hanno apportato un considerevole contributo allo sviluppo di applicazioni Web sicure ed efficienti. Si tratta di una tecnica basata sul linguaggio Java che garantisce una straordinaria riusabilità e demanda le poliitiche di gestione e sicurezza ad un Application Server. L'utilizzo delle JSP in ambito business è ormai talmente consolidato da rappresentare uno standard. Ivan Venuti ci illustra le fondamenta della tecnologia per poi proseguire nella descrizione degli aspetti avanzati. Il libro è dedicato a chi vuole imparare, ma anche a chi avendo una conoscenza di base delle JSP ne vuole approfondire alcuni aspetti.

la tecnologia da conoscere per sviluppare applicazioni web per le aziende

- **Cosa sono le JSP e come funzionano**
- **Installazione di Tomcat e servizi di base**
- **Strutture predefinite e sintassi**
- **Aspetti avanzati, dai Tag a JDBC**

GLI ALLEGATI DI IOPROGRAMMO

▼ Arrivano anche i video...

In collaborazione con MSDN, *Microsoft Developer Network,* nasce una nuova iniziativa per consentire agli sviluppatori di migliorare le proprie conoscenze di sviluppo seguendo dei minicorsi d'aggiornamento seduti comodamente davanti al proprio computer quando e come lo desiderano. I WebCast di MSDN rappresentano un'ottima opportunità per approfondire le tecnologie di sviluppo Microsoft. Si tratta di eventi online interattivi e gratuiti di 60/90 minuti tenuti dagli esperti in tecnologia Microsoft in collaborazione con i partner di formazione MSDN. ioProgrammo ha voluto fornire un ulteriore servizio ai programmatori ovvero quello di poter seguire i corsi di formazione indipendentemente dalla programmazione reale con cui essi vengono tenuti OnLine. Avendoli a disposizione sul CDRom possono essere seguiti in un qualunque momento della giornata, e magari consultati per rivedere un passaggio poco chiaro o per ottenere informazioni su una certa tecnica. Siamo ben lieti di aver dato luogo a quesa iniziativa in stretta collaborazione con Microsoft ed aspettiamo il vostro feedback.

I VIDEOCORSI PER PROGRAMMARE BENE

5 WEBCAST UFFICIALI MICROSOFT

IN ESCLUSIVA GRATIS NEL CD "I CORSI DI FORMAZIONE" DA SEGUIRE COMODAMENTE SUL PC

msdn WEBCAST

VISUAL STUDIO.NET

- Le guideline per sviluppare applicazioni ben strutturate
- Le DEMO per usare subito le novità di Visual Basic.NET

ASP.NET 2.0

- L'architettura e i primi passi per programmare il web
- Massimo riutilizzo con le Master Pages e i temi
- Data Access e Object Binding, come usare i database

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN

Come è composto tipicamente un Webcast?

Normalmente vengono illustrate una serie di Slide commentate da un relatore. A supporto di queste presentazioni vengono inserite delle Demo in presa diretta che mostrano dal vivo come usare gli strumenti oggetto del Webcast

Come mai trovo riferimenti a chat o a strumenti che non ho disponibili nei WebCast allegati alla rivista?

La natura dei WebCast è quella di essere seguiti OnLine in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. In questa ottica è possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi etc. I WebCast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i WebCast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei WebCast MSDN direttamente da CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con tutti gli strumenti possibili.

Su ioProgrammo troverò tutti WebCast di Microsoft?

Ne troverai sicuramente una buona parte, tuttavia per loro natura i webcast di Microsoft vengono diffusi anche OnLine e possono essere seguiti previa iscrizione. L'indirizzo per saperne di più è: http://www.microsoft.it/msdn/webcast_msdn. segnalo nei tuoi bookmark. Non puoi mancare.

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente si.

News

PC A BASSO PREZZO PER LENOVO

Lenovo Group erede del comparto Hardware di IBM ha appena lanciato sul mercato una serie di PC a costi veramente molto contenuti. In particolare sono previsti 349$ per un Desktop e 599$ per un laptop. La nuova linea di prodotti sarà divisa in due linee di produzioni, la C e la J la prima equipaggiata con processori Intel, la secondo con processori AMD. Lenovo in questo modo reagisce alla crescente offerta di Dell che proprio recentemente aveva ottenuto grandi successi proponendo sistemi dai costi veramente ridotti. Tutte e due le case puntano comunque sulla vendita OnLine che riduce notevolmente i costi dovuti alla distribuzione.

WINDOWS 98 E WINDOWS ME ADDIO

È così siamo giunti alla fine di un'epoca. Microsoft ha ufficialmente annunciato che dal 30 Giugno 2006 cesserà il supporto Windows 98 e Windows Millennium. Il consiglio di MS è quello di migrare almeno a Windows XP. Se non ci impressiona il fatto che Windows ME sia stato lasciato infine abbandonato a se stesso, confessiamo che un minimo di riflessione è dovuta almeno a Windows 98 che per anni è stato il glorioso portabandiera delle tecnologie MS e che ancora oggi probabilmente fa girare un discreto numero di processori.

E' MISTERO SU ORIGAMIPROJECT

Fra le nuove registrazioni di domini internazionali spicca questo origamiproject.com, intestato addirittura ad Microsoft Corporation. Al momento in cui scriviamo visitando il sito si accede a un' animazione in flash che proietta le seguenti voci: "Do you know me? - Do you know what i can do?
and where i can go?" su cosa sia e cosa voglia rappresentare questo nuovo progetto di Microsoft è mistero fitto.

WINDOWS VISTA FA SEI!

Sarebbero sei le versioni programmate da Microsoft, basate sull'annunciata nuova release di Windows: nome in codice Vista.

Tre di queste sarebbero progettate per soddisfare le esigenze degli utenti domestici, due per le aziende e una dedicata ai nuovi mercati. In particolare le sei versioni in questione sono state indicate come: Vista Business, Vista Enterprise,Vista Home Basic, Vista Home Premium, Vista Ultimate, Vista Starter. Le due versioni Business ed Enterprise sarebbero pressoché identiche e destinate ad un utenza decisamente aziendale, equipaggiate con strumenti pensati per aiutare l'utente a mantenere ordinata la presenza di documenti all'interno delle proprie macchine e nella lan aziendale, tuttavia la versione Enterprise includerebbe un supporto decisamente importante alla criptazione dei documenti. È quindi prevista una certa attenzione alla privacy e alla sicurezza, temi che recentemente stanno muovendo e non di poco il mercato informatico.

La versione Home Basic includerebbe invece soltanto strumenti semplificati per coloro che utilizzano il PC per effettuare operazioni ordinarie come navigare su internet, leggere la posta oppure creare semplici documenti. E' prevista in questa versione una stoccata al nemico giurato: "Google", sembrerebbe infatti che sarà incluso uno strumento per la ricerca delle informazioni molto simile al Desktop Search Bar promosso dal rivale. Anche in questo caso il sistema manterrà un occhio di riguardo verso la sicurezza includendo un firewall leggermente più sofisticato di quello attuale. Vista Home Premium manterrà tutte le caratteristiche della versione precedente ma sarà equipaggiata con una nuova interfaccia grafica chiamata Aero. Questa interfaccia è pensata per aiutare gli utenti nel gestire documenti

ORACLE ACQUISTA SLEEPYCAT

Oracle ha appena annunciato di avere acquisito SleepyCat, software che fino a ieri deteneva i diritti sul famoso DB. Berkeley DB. L'annuncio è importante in un panorama, quello dei database, sempre più mobile e caratterizzato da una concorrenza spietata. L'acquisto di Berkeley DB rappresenta per Oracle la possibilità di fornire un prodotto performante, completo e utilizzabile anche a livello enterprise a prezzi decisamente contenuti. Le dichiarazioni di Mike Olson CEO di Sleepycat Software sono significative: " Siamo molto entusiasti di poter portare il nostro prodotto sul mercato enterprise e di entrare a far parte di una delle maggiori compagnie nel settore dei DB al mondo. I prodotti di Sleepycat, la base dei clienti e un modello di business ormai sperimentato coniugate alla larga esperienza di Oracle oltre che ovviamente alla tecnologia e alle risorse di questa compagnia, ci consentiranno di servire al meglio i nostri clienti e supportare maggiormente la community OpenSource". Secondo gli esperti il mercato dei database Embedded nel 2005 ha raggiunto un fatturato di due miliardi di dollari e si prospetta che raggiungere 3.2 milioni di dollari nel 2009. L'acquisto di SleepyCat diventa perciò strategicamente importante per Oracle che piazza un nuovo prodotto all'interno del mercato dei DB embedded e si proietta in pole position per diventare leader di questo crescente segmento anche in futuro.

multimediali come film e musica e inoltre dovrebbe includere alcuni strumenti per la masterizzazione e l'authoring dei cd/dvd. Una nota importante è relativa alla capacità di questa nuova versione di poter interagire con una console Xbox 360. Infine Vista Ultimate e Vistar starter Kit rappresentano i due poli opposti dell'emergente tecnologia, da un lato Ultimate conterrà tutti gli strumenti delle versioni business ed home condensate in un'unica installazione, viceversa la versione Starter sarà un edizione limitata pensata per soddisfare le esigenze di paesi che non dispongono ancora di strumenti abbastanza potenti da poter gestire un sistema così complesso.

SUN ORGANIZZA L'AJAX DAY

Ajax è decisamente una tecnologia che sta facendo parlare di se. La possibilità di aggiornare solo parti di una pagina dinamica senza dover ricorrere all'uso dei frame ma considerando ogni oggetto della pagina come un elemento a se stante, apre scenari innovativi nel campo dello sviluppo web.
Sono in tanti a puntare su questa nuova tecnologia, ed anche se ancora fa capolino specialmente nelle Intranet, mentre ancora sono pochi i siti web che sono stati convertiti, si intravede la possibilità di una rete decisamente diversa per il futuro. Di tutto questo è ben cosciente Sun che ha organizzato a Milano e a Roma per il sei e il sette Marzo una giornata di seminari tesa ad illustrare appunto le nuove tecnologie. Parallelamente all'introduzione alla nuova tecnologia, SUN illustra anche le capacità di JAVA Studio Creator 2 in relazione proprio allo sviluppo di web application con AJAX.
L'iniziativa si è rivelata sicuramente interessante, sia per coloro che non utilizzano strumenti legati a Java ma che vogliono imparare a usare AJAX sia per coloro che sono già esperti di tecnologie SUN e vogliono approfondire il legame con strumenti avanzati quali Studio Creator 2. In ogni caso è da apprezzare lo sforzo di Sun nel volere sempre adottare le tecnologie più utili e innovative per gli sviluppatori

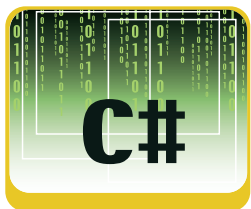
YAHOO LANCIA IL PHP CENTER

Yahoo è fra le prime grandi compagnie ad offrire uno spazio/community per PHP. E' recente infatti l'annuncio dell'apertura di una sezione del portale di Yahoo proprio dedicata a PHP. L'indirizzo è http://developer.yahoo.net/php. Questo nuovo spazio assumerà la forma di una community. Nelle dichiarazioni di Jeffery Mcmanus la motivazione appare forte e delicata, ovvero quella di offrire una piattaforma universale per lo sviluppo di applicazioni che possano in qualche modo essere integrate con i servizi già offerti dal l'azienda fondata dai due ragazzi terribili David Filo e Jerry Yang. Il legame fra PHP e Yahoo è d'altra parte noto. Fra i dipendenti di yahoo si conta anche Rasmus Lerdorf.

MYSQL COMPRA NETFRASTRUCTURE INC

È proprio il mercato dei database a farla da padrone nell'intricato mondo dello sviluppo in questo inizio di 2006, così mentre Oracle si lancia alla conquista del mercato embededd, arriva l'annuncio di MySQL AB sempre più leader nel segmento Web. MySQL ha appena questiato Netfrastructure INC. Una software house sufficientemente piccola ma con una grande esperienza proprio nello sviluppo di Web Application, e che tra i suoi fondatori annovera un nome importante per quanto riguarda la programmazione di architetture di database, ovvero JIM Starkey, personaggio piuttosto noto in questo settore. In particare Starkey e la Netfrastrucure INC. avrebbero fornito un certo impulso allo sviluppo di Interbase e Firebird. Il primo dei due prodotti è finito poi nelle mani di Borland, il secondo ha continuato ad essere sviluppato in modo Free. In tutti e due i casi i database in questione hanno apportato innovazioni significative in questo settore.

CREIAMO UN GIOCO: TIRO AL BERSAGLIO

GRAZIE AL RILASCIO DELLA SUA LIBRERIA OPEN SOURCE COMPUTER VISION LIBRARY (OPENCV), INTEL CORPORATION HA ACCELERATO LO SVILUPPO DI APPLICAZIONI IN GRADO DI SFRUTTARE LA WEBCAM. VEDIAMO COME.

In questo articolo ci divertiremo a creare un piccolo gioco di tiro al bersaglio elettronico. Il funzionamento sarà semplice. Avremo bisogno di un foglio di carta su cui stampare sette cerchi concentrici e che rappresenterà il nostro bersaglio, di una webcam da posizionare dietro il bersaglio, e di un puntatore laser. Niente paura, il puntatore laser può essere tranquillamente una di quelle luci a fascio stretto che viene utilizzata spesso come portachiave. Volendo divertirvi potete anche usare un semplice condensatore per rendere l'effetto molto simile a quello ottenuto da una pistola laser.
Il funzionamento tecnico del gioco è abbastanza semplice. L'utente spara verso il bersaglio con il laser, la webcam posta dietro il bersaglio intercetta il fascio di luce e attribuisce un punteggio basato sulla precisione del tiro sul bersagno.

COME FARE?

Utilizzeremo una libreria sviluppata da Intel Corporation sotto licenza OpenSource, la OpenCv sufficientemente potente per aiutarci al raggiungimento dei nostri scopi. Il codice sorgente di tutte le funzioni contenute nella libreria sono scritte in C e la ridistribuzione del codice sorgente è priva di royalty. L'utilità di questa libreria è innegabile nel campo dell'aiuto ai disabili o per migliorare l'interazione uomo-macchina. Ma può essere usata anche per impieghi, come dire, più "ludici".
Possiamo pensare di realizzare su di essa un'applicazione con la quale calcolare il punteggio di ogni tiro al target, analizzando il flusso video prodotto dalla videocamera digitale per individuare con una buona approssimazione dentro quale cerchio sta ricadendo il punto del laser. In sostanza, per realizzare questo tiro al bersaglio abbiamo bisogno di uno strato software, che svolga la funzione elaborativa di cui spiegheremo i principi di funzionamento, e un po' di hardware veramente comune che non richiede una spesa da capogiro: una webcam via USB a bassa risoluzione (320x120 a 15 €) e un laser di quelli che si acquistano dagli ambulanti per strada (7-10 €). Totale 22-25 €.
Naturalmente bisogna aggiungere il "target", il disegno di un certo numero di cerchi concentrici.

ANALISI

Preso così il compito che ci siamo prefissati è veramente arduo a causa del numero di variabili che dovremmo prendere in considerazione; le principali sono: la posizione della webcam rispetto all'obiettivo, il calcolo della rotazione degli assi visivi, la diversa luminosità del laser e della dimensione del punto prodotto.
Per semplificare il progetto partiamo da due semplici ipotesi:

- il target è prefissato: è costituito da 7 cerchi cerchi concentrici dove, preso come riferimento il raggio r del cerchio più piccolo , il secondo cerchio è 2r, il terzo 3r e così via.

- Il foglio su cui viene stampato il disegno viene appeso di fronte alla webcam in modo che sia abbastanza dritta.

In generale individuiamo due macroattività, nella prima calibreremo la webcam in modo da individuare la posizione del disegno e ricostruire su di esso il modello del target; la seconda avrà la funzione di catturare la presenza del punto luce del laser sull'obiettivo e

posizionarlo rispetto ai cerchi concentrici in modo da assegnare il giusto punteggio.

FASE UNO: INDIVIDUARE IL MODELLO

Dentro la distribuzione di OpenCv troviamo *CvCam*, un modulo universale e multi piattaforma messo a disposizione dall'Intel per la gestione dei flussi video ottenuti attraverso l'uso di videocamere digitali. Essa è implementata come una *Dynamic Link Library (DLL)* per le piattaforme Windows e come una *Shared Object Library (so)* per i sistemi Linux. *CvCam* fornisce un'interfaccia di funzioni *(API)* mediante le quali è possibile leggere e controllare lo stream video, elaborare ogni singolo frame individuate all'interno del flusso di dati e renderizzare il risultato di tale elaborazione. Per semplificare un po' la mole di lavoro sfrutteremo quindi le funzionalità messe a disposizione da *CvCam*. In particolare, utilizzeremo questo modulo per individuare la webcam tra tutti i dispositivi collegati al computer e inizializzarla in modo da prepararla a fornirci le immagini riprese. Una volta pronta, dalla webcam possiamo recuperare direttamente lo stream video lavorando immagine per immagine.

INIZIALIZZARE LA WEBCAM

La procedura di inizializzazione della WebCam è abbastanza semplice, anche perché ogni ingrato compito di interfacciamento con il dispositivo è delegato alla libreria OpenCV. Le istruzioni da richiamare sono veramente poche. La prima cosa da fare è ottenere il numero di webcam collegate con il computer

```
int numCamere=cvcamGetCamerasCount();
```

Verificata la presenza di almeno un dispositivo compatibile, si procede dando la possibilità all'utente di selezionare quale camera vuol utilizzare

```
int* out;
int nselected = cvcamSelectCamera(&out);
```

usando *cvcamSelectCamera()* si aprirà una finestra come quella mostrata in **Figura 1**.
È possibile notare che OpenCv dà la possibilità di selezionare due diverse webcam.
Questo perché la libreria è nata per applicazioni

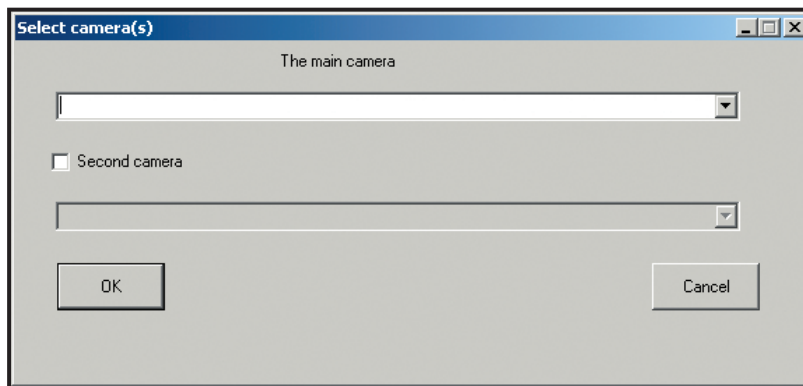



Fig. 1: *Qui è possibile scegliere la webcam tra quelle individuate in maniera automatica*

"stereoscopiche", cioè dove è necessario l'uso di una coppia di telecamere per ricostruire un modello reale di ciò che si sta riprendendo. L'impiego di due videocamere digitali consente di ottenere informazioni sulla profondità e sull'immagine altrimenti non possibile limitandoci ad una sola, in questo modo è possibile realizzare una vasta gamma di applicazioni di computer vision, tra cui il riconoscimento gestuale, la registrazione di oggetti e il riconoscimento facciale. Ottenuta la scelta dell'utente si procede con l'inizializzazione della webcam

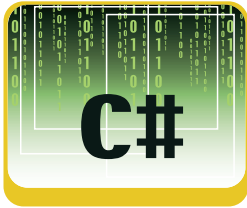
```
int desiredcamera = out[0];//for example
cvcamSetProperty(desiredcamera,
                    CVCAM_PROP_ENABLE, CVCAMTRUE);

CvCapture* capture =cvCaptureFromCAM( out[0] );
```

A questo punto la webcam, virtualmente rappresentata dall'entità capture, è pronta a rispondere ad ogni nostro comando. Possiamo vedere ad esempio come è possibile ottenere quello che sta riprendendo in un determinato istante:

```
cvGrabFrame( capture );
IplImage* input_img= cvRetrieveFrame( capture );
```

FILTRI DIGITALI

Tra le funzioni avanzate più frequentemente usate in Computer Vision ci sono i filtri digitali, il cui effetto è uguale a quello dei corrispondenti filtri ottici usati in fotografia. Le funzioni matematiche dei filtri digitali sono implementate usando la convoluzione spaziale 3x3 dove il valore di ogni pixel è rimpiazzato con la media algebrica ponderata del suo valore e del valore degli 8 pixels adiacenti: i valori dei 9 pixels vengono moltiplicati per il valore dei corrispettivi pesi, poi viene eseguita la media il cui risultato va a sostituire il valore del pixel centrale. Questa operazione viene ripetuta per tutti i pixels dell'immagine o di un'area selezionata in cui si vuole eseguire il filtraggio.

La funzione *cvGrabFrame()* comanda il device a “scattare” un’istantanea di quello che sta riprendendo in quel preciso momento. Per poter ottenere l’immagine catturata si ricorre a *cvRetrieveFrame()*. Con pochi e semplici comandi abbiamo quindi preso il controllo della webcam. Dal flusso video vengono così isolate delle singole immagini su cui possiamo adesso effettuare tutte le elaborazioni di cui abbiamo bisogno. Cominciamo con la calibratura della webcam, necessaria per individuare la posizione del target sull’immagine e costruire su di essa il nostro modello.

CENTRARE IL MODELLO

Posizioniamo il device in modo da riprendere per intero il disegno a cerchi concentrici che costituisce il target da individuare. La calibratura consiste nel sottoporre ogni singola immagine ripresa ad una serie di elaborazioni il cui scopo è quello di individuare la presenza di figure geometriche il più possibile simili a cerchi. In generale basterebbe estrarre il contorno dall’immagine (operazione tra l’altro molto semplice usando la funzione cvFindContours() con opportuni parametri) e passarlo come ingresso a cvFitEllipse(). Beh, qui cominciamo ad incontrare le prime difficoltà. Infatti, non possiamo illuderci di riuscire ad ottenere il ricalco del contorno di tutte le figure individuabili: le webcam di solito hanno una bassa risoluzione e sono molto sensibili alla variazione di luce, creando di fatto delle aree a diversa luminosità su ciascuna immagine che non risulterà quindi omogenea. In altre parole ogni frame è soggetto ad una forte dose di disturbo che influisce in maniera molto significativa sul risultato della cvFindContours(): se non mettessimo dei vincoli a questa operazione otterremmo anche il contorno di questi disturbi il che impedirebbe di fatto ogni elaborazione successiva. Dobbiamo cercare di eliminare questi “rumori di fondo” dall’immagine operando una serie di filtraggi. Per prima cosa convertiamo l’immagine a colori [**Figura 2**] portandola in toni di grigio [**Figura 3**], in modo da ridurre la quantità di informazioni da elaborare; oltretutto l’utilizzo di ciascun canale di colore RGB darebbe soltanto risultati ridondanti.

```
IplImage* input_img=...
IplImage*m_tmp8uC1T=...
cvCvtColor(input_img, m_tmp8uC1T,
                                CV_RGB2GRAY);
```

Fig. 2: ***Questo è ciò che riprende la telecamera***

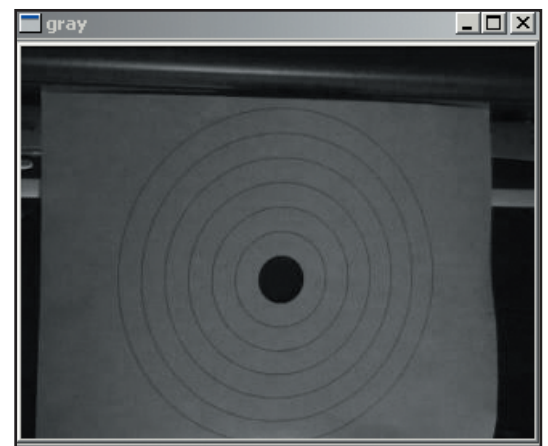



Fig. 3: ***L’immagine originale viene convertita in toni di grigio***

Come si può vedere, mettendo a confronto le due immagini, la seconda è ruotata di 180 gradi, ma non è un problema, basta tenerne conto nel proseguio. La funzione cvCvtColor() converte l’immagine in ingresso da uno spazio di colore ad un altro. Il primo parametro è l’immagine sorgente, il secondo è quella risultante la trasformazione mentre il terzo corrisponde al tipo di conversione da adottare. In particolare è il tipo di trasformazione è una costante intera codificata come CV_<src_color_space>2 <dst_color_space>. A questo punto l’immagine è pronta per essere resa omogenea mediante un filtro di smooth: l’effetto è che l'immagine viene come "piallata", le tonalità di grigio sfumati riducendo l'effetto macroblocco [**Figura 4**].

```
cvSmooth( m_tmp8uC1T, image05, CV_BLUR,3,3,0);
```

Adesso applichiamo l’operatore Canny, il quale evidenzia ed isola il filo degli oggetti presenti sull’immagine [Fig. 5]; per capirci meglio, è un po’ quello che facevamo da piccoli quando mettevamo un foglio bianco so-

NOTE

FILTRO CANNY

L’operatore Canny consiste in un filtro digitale di edge-detection sulle immagini digitali al fine di identificare ed estrarre dei profili dagli elementi scelti. Il profilo ricavato mediante tale tecnica viene poi usato per la registrazione di un'immagine su un sistema di riferimento. L’uso di questo filtro è sempre il primo passo nelle operazioni di identificazioni di oggetti e figure all’interno delle immagini. Queste devono prima essere sottoposte ad una fase di pre-elaborazione in modo da renderle adette all’applicazione dell’operatore Canny.

Fig. 4: ***Si applica il filtro smooth per eliminare quel fastidioso effetto macroblocco***

pra un disegno e con una matita ricalcavamo i contorni che apparivano come un'ombra.

```
cvCanny(m_tmp8uC1T,image03, 50, 100, 3);
```

l'algoritmo di Canny lavora su alcuni parametri, giocando con questi valori si può ridurre ulteriormente l'effetto del disturbo (che tra l'altro dipende dalla webcam).

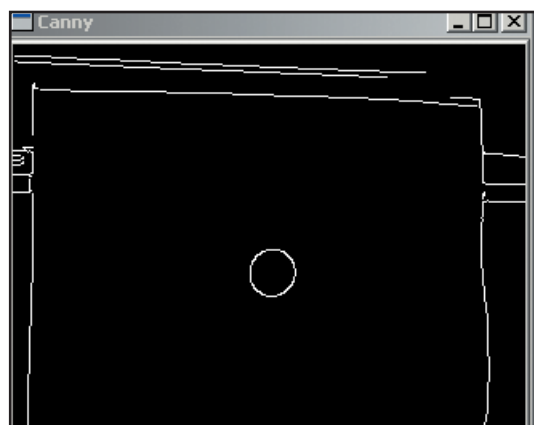



Fig. 5: ***Risultato dopo l'operazione di Canny***

In *image05* abbiamo finalmente un'immagine accettabile, privata sostanzialmente da tutti i disturbi e contenente soltanto l'informazione di cui avevamo bisogno, anche se in effetti è stata privata praticamente di tutto! Soltanto una cosa è stata risparmiata da questo filtraggio aggressivo sull'immagine catturata dalla webcam: il cerchio centrale del target, quello tutto nero. Il passo successivo adesso è quello di estrarre il contorno, mediante cvFindContours(), dall'immagine così ottenuta

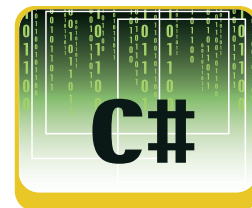
```
CvMemStorage* stor;
CvSeq* cont;
CvBox2D32f* box;
CvPoint* PointArray;
```

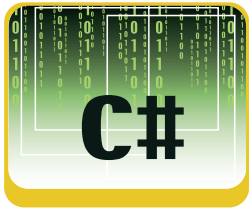
```
CvPoint2D32f* PointArray2D32f;
stor = cvCreateMemStorage(0);
cont = cvCreateSeq(CV_SEQ_ELTYPE_POINT,
                    sizeof(CvSeq), sizeof(CvPoint) , stor);

cvFindContours( image03, stor, &cont,
                    sizeof(CvContour), CV_RETR_EXTERNAL,
                                    CV_CHAIN_APPROX_NONE);
```

in cont c'è adesso tutta la sequenza di punti che individuano l'estremità dei segmenti che costituiscono il contorno in generale. La sequenza è in realtà suddivisa per blocchi, tenendo conto che il contorno individuato non è proprio perfetto e ci sono zone prive di informazioni necessarie per renderlo continuo. Nonostante questo possiamo procedere lavorando per ciascun blocco, estraendo i punti che lo costituiscono e cercando l'eventuale ellisse che li collega tutti mediante la funzione cvFitEllipse (). Per individuare correttamente un ellisse sono necessati almento 5 punti: per essere sicuri escluderemo tutti i blocchi con meno di 10 punti.

```
for(;cont;cont = cont->h_next){
 int i;
 //ritorna il numero totale di punti del blocco
 int count = cont->total;
 CvPoint center;
 CvSize size;
 double area, perimeter, roundness;
 if( count < 10 )     continue;
 // Allochiamo la memoria per contenere tutti i punti
 PointArray = (CvPoint*)malloc(
                              count*sizeof(CvPoint) );
 PointArray2D32f= (CvPoint2D32f*)malloc(
                          count*sizeof(CvPoint2D32f) );

 // Allochiamo la memoria che conterrà le
informazioni relative al box
 box = (CvBox2D32f*)malloc(sizeof(CvBox2D32f));
```

CHE COSA È IL TIPO SYSTEM.INTPTR

È un tipo specifico per la piattaforma utilizzato per rappresentare un puntatore o un handle. IntPtr è progettato per essere un integer di una dimensione specifica per la piattaforma. È previsto che un'istanza di questo tipo sia a 32 bit per hardware e sistemi operativi a 32 bit e a 64 bit per hardware e sistemi operativi a 64 bit. È possibile utilizzare il tipo IntPtr in linguaggi che supportano i puntatori e come elemento comune per riferimenti ai dati in linguaggi che supportano i puntatori e in linguaggi che non li supportano. È inoltre possibile utilizzare gli oggetti IntPtr per contenere gli handle. Istanze di IntPtr, ad esempio, vengono spesso utilizzate nella classe System.IO.FileStream per contenere handle di file o nel nostro caso, per recuperare un bitmap in memoria generata tramite la dll.

```
cvCvtSeqToArray(cont, PointArray,
                                    CV_WHOLE_SEQ);
for(i=0; i<count; i++)
{
 PointArray2D32f[i].x = (float)PointArray[i].x;
 PointArray2D32f[i].y = (float)PointArray[i].y;
}
//Come risultato da un box
cvFitEllipse(PointArray2D32f, count, box);
// Calculaliamo l'area e il perimetro
area = cvContourArea(cont, CV_WHOLE_SEQ);
perimeter = cvArcLength(cont, CV_WHOLE_SEQ,
                                              1);
roundness = (4 * 3.14159 * area) / (perimeter *
                                      perimeter);
center.x = cvRound(box->center.x);
center.y = cvRound(box->center.y);
size.width = cvRound(box->size.width*.5);
size.height = cvRound(box->size.height*.5);
box->angle = -box->angle;
if (roundness > .875) // perfect circle = 1. goes
                                       down to 0.
{if (size.width > 5)
  {
     ...//trovato!
  }
  centro=center;
  raggio=max(size.width, size.height);
}
...
}
```

Calcolando il perimetro del blocco individuato e l'area, possiamo valutare se quello che abbiamo trovato è qualcosa di simile ad un cerchio calcolando il parametro Roundness, cioè il rapporto tra l'asse maggiore e quello minore dell'ellisse: se questo rapporto è uguale a 1, la figura è un cerchio perfetto, 0.875 è un buon compromesso dato che per essere perfetto la webcam dovrebbe centrare esattamente il target.
In fin dei conti era questo di cui avevamo bisogno, la posizione del centro (box->center) e il raggio "base" (metà del massimo tra i due lati del box) , cioè la distanza tra un anello ed il successivo che per impostazione è sempre lo stesso. Sia dr il raggio del cerchio centrale, quello successivo sarà 2dr, il seguente 3dr e così via. Il target è quindi completamente individuato e modellizzato. Come abbiamo visto, una tecnica di ricerca così semplice richiede che la webcam sia centrata sul target per avere un risultato ottimale. Comunque l'errore introdotto dalla posizione della videocamera può essere reso minimo spostandola finché il modello ricostruito sia il più vicino possibile con il target vero e proprio. Come fare? Basta visualizzare l'immagine originale e disegnarci di sopra i modello a cerchi ricostruito!

```
cvNamedWindow("Target",1);
```

apre una finestra in cui visualizzare l'immagine

```
for (int i=1; i<=7; i++)
 cvCircle(input_img, centro, i*raggio,
                         CV_RGB(255,255,255), 1);
cvShowImage("Target", input_img);
```

FASE DUE: LA POSIZIONE DEL LASER

Abbiamo il modello e quindi la prima macrooperazione è conclusa. Adesso è necessario individuare la posizione del punto prodotto dal laser sul target. Beh, sfruttiamo la stessa tecnica adottata per la calibrazione, con un particolare in più. Perché? Beh, il punto è rilevabile come un cerchio e quindi basterebbe di nuovo trovare tutte le figure che più si avvicinano ad un cerchio. Ma come abbiamo visto c'è molto errore sull'immagine acquisita dalla webcam e comunque il punto del laser non è ben visibile, appare come un alone senza bordi netti.
Come fare allora? Per prima cosa eliminiamo ogni movimento rilevabile intorno al modello ricostruito, isolando l'area dell'immagine occupata dal rettangolo che circoscrive il target individuato.

```
cvSetImageROI( image, cvRect(oldCenter.x-
          7*oldRaggio, oldCenter.y-7*oldRaggio,
                   14*oldRaggio,14*oldRaggio));
```

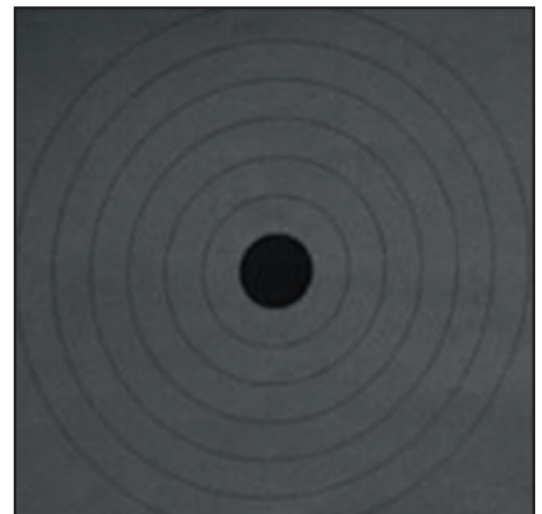



Fig. 6: *Immagine di confronto*

NOTE

IPLIMAGE

IplImage (strano nome che sta a significare Image Processing Library – Image) è il formato standard usato nelle API IPL e OpenCV dell'Intel per rappresentare un'immagine. Tramite questa struttura è possibile gestire completamente tutto ciò che riguarda le immagini: caricamento e salvataggio da file di tutti i formati (jpeg, gif, tiff, etc), elaborazione e visualizzazione a finestra.

Fig. 7: *Istantanea che riprende il colpo del laser*

Fatto questo, mettiamo a confronto due diverse immagini, la prima [**Figura 6**] che riprende il target senza il dot (punto del laser sul foglio) e l'altra con il dot [**Figura 7**].

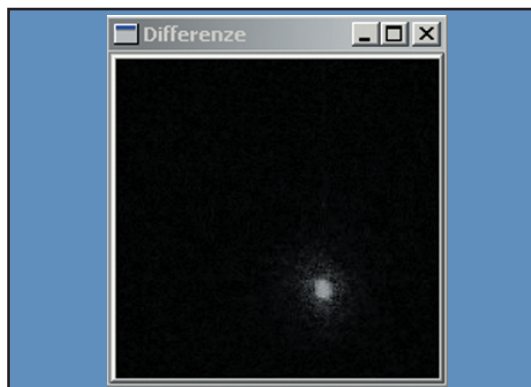



Fig. 8: *Differenza tra la Fig. 6 e la Fig. 7*

Possiamo evidenziare ed isolare la differenza tra le due istantanee [**Figura 8**], che è proprio l'area dove ricade il dot se la webcam e il target sono mantenuti fermi,

```
cvAbsDiff( oldImage, imageB, imageRis );
```

e poi procedere per individuare il centro dell'area riportandolo in coordinate relative al centro del modello ricostruito [**Figura 9**].

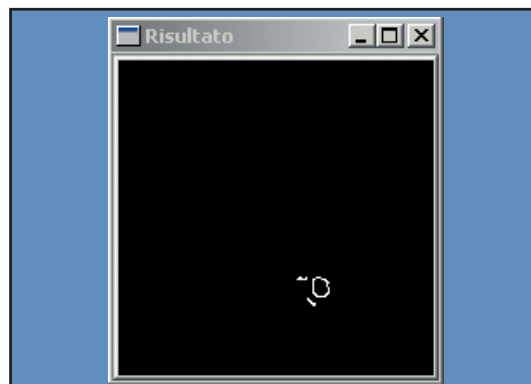
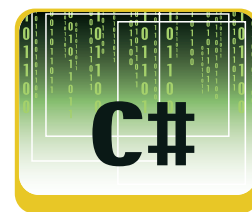



Fig. 9: *Dopo l'operazione di Canny.*

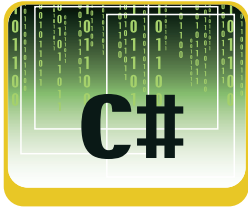
DA IPLIMAGE A BITMAP

L'IplImage è una struttura proprietaria dell'Intel per rappresentare i dati che costituiscono un'immagine. Per poter consentire l'interoperabilità con altre applicazioni è possibile rendere disponibile l'immagine incapsulata nella IplImage senza troppo impegno. Possiamo infatti definire la funzione IplImageToHBitmap() che converta una IplImage passata in ingresso in una Bitmap (HBITMAP).

```
...
HBITMAP IplImageToHBitmap(IplImage* image)
{
   HBITMAP hBitmap=CreateBitmap(
                    image->width,image->height,image
                                  ->nChannels*8,1);
  long p=SetBitmapBits(hBitmap,image->
                         imageSize,image->imageData);
  return hBitmap;
}
...
```

COSA È UNA DLL

DLL è l'acronimo di Dynamic Link Library ovvero in italiano Libreria a Collegamento Dinamico. Esse possono essere viste come una raccolta di funzioni richiamabili da qualsiasi programma in grado di interfacciarvisi. Di solito all'interno delle DLL si introducono delle istruzioni da condividere o di interfacciamento, consentendo così di non inserire codice duplicato in applicazioni differenti.
Tramite le DLL è anche possibile esportare delle funzionalità sviluppate per un dato linguaggio di programmazione verso altri.Si possono assimilare le DLL come a unità che contengono codice eseguibile collegabile in fase di esecuzione.

Tale funzione costruisce la bitmap tramite una chiamata al metodo *CreateBitmap()* usando i parametri della IplImage e vi copia dentro la sequenza di dati che la compongono

```
...
HBITMAP CreateBitmap (int w,int h,WORD bpp,int
                                          nSize)
{
HDC hDC = ::CreateCompatibleDC(0);
BYTE tmp[sizeof(BITMAPINFO)+255*4];
BITMAPINFO *bmi = (BITMAPINFO*)tmp;
HBITMAP hBmp;
int i;
memset(bmi,0,sizeof(BITMAPINFO));
bmi->bmiHeader.biSize =
                          sizeof(BITMAPINFOHEADER);
bmi->bmiHeader.biWidth = w;
bmi->bmiHeader.biHeight = h;
```

```
bmi->bmiHeader.biPlanes = nSize;
bmi->bmiHeader.biBitCount = bpp;
bmi->bmiHeader.biCompression = BI_RGB;
bmi->bmiHeader.biSizeImage = w*h*1;
bmi->bmiHeader.biClrImportant =0 ;
switch(bpp)
{
  case 8 :
  for(i=0 ; i < 256 ; i++)
  {
   bmi->bmiColors[i].rgbBlue = i;
   bmi->bmiColors[i].rgbGreen= i;
   bmi->bmiColors[i].rgbRed= i;
  }
  break;
  case 32:
  case 24:
  ((DWORD*) bmi->bmiColors)[0] = 0x00FF0000; /*
                                componente rossa*/
  ((DWORD*) bmi->bmiColors)[1] = 0x0000FF00; /*
                                componente verde*/
  ((DWORD*) bmi->bmiColors)[2] = 0x000000FF; /*
                                componente blu*/
  break;
}
hBmp =::CreateDIBSection(
                   hDC,bmi,DIB_RGB_COLORS,NULL,0,0);
::DeleteDC(hDC);
return hBmp;
}
…
```

SUL WEB

Per informazioni sulla libreria Intel OpenCV fare riferimento al sito web ufficiale:
http://www.intel.com/technology/computing/opencv/index.htm

In maniera analoga possiamo procedure per ottenere una IplImage a partire da una HBIT-MAP

```
…
IplImage* HBitmapToIplImage(HBITMAP hBmp)
{
BITMAP bmp;

::GetObject(hBmp,sizeof(BITMAP),&bmp);

int nChannels = bmp.bmBitsPixel == 1 ? 1 :
                          bmp.bmBitsPixel/8 ;
int depth = bmp.bmBitsPixel == 1 ? IPL_DEPTH_1U
                                 : IPL_DEPTH_8U;

IplImage* img = cvCreateImageHeader(
              cvSize(bmp.bmWidth, bmp.bmHeight)
                         , depth, nChannels );

img->imageData =(char*)malloc(
              bmp.bmHeight*bmp.bmWidth*nChan
                          nels*sizeof(char));
memcpy(img->imageData,(char*)(
             bmp.bmBits),bmp.bmHeight*b
               mp.bmWidth*nChannels);
return img;

}
…
```

Questa funzione recupera l'oggetto BITMAP rappresentato dall'HBITMAP, costruisce una entità IplImage e vi copia dentro la sequenza di bit che costituisce l'immagine.

CAMLIBRARY

Per semplificare l'interfacciamento con la libreria OpenCV costruiremo una dll con Visual C++ (vedi box) con la quale metteremo a disposizione una serie di comandi per agire sul device e per recuperare le informazioni dall'immagini via via prodotte.

- **bool InitCamera():** inizializza la webcam
- **bool GrabFrame():** effettua uno scatto
- **HBITMAP RetriveFrame():** ritorna l'indirizzo della bitmap
- **bool Find_Circles():** cerca il centro e ritorna il modello
- **bool Find_Shot():** determina la presenza e la posizione del dot
- **int Centro_X():** ritorna la coordinata X del centro del modello
- **int Centro_Y():** ritorna la coordinata Y del centro del modello
- **int Centro_R() :** è il raggio base del modello a cerchi
- **int Shot_X():** la coordinata X del dot
- **int Shot_Y():** la coordinata Y del dot

Per esportare le funzioni in modo da condividerle con un'applicazione di gestione, dobbiamo definire l'intestazione di ciascuna funzione nel modo seguente

```
extern "C" __declspec(dllexport) bool __stdcall
                              InitCamera(){…}
```

Le keyword extern "C" __declspec(dllexport) sono necessarie per indicare al compilatore che la funzione InitCamera() deve essere esportata e quindi deve essere resa visibile all'esterno e il nome della funzione deve essere creato utilizzando il linking di tipo "C" (export "C").
Si poteva usare anche il linking normale di tipo C++ ma il compilatore avrebbe creato per la funzione, a causa della name decoration diversa, un nome strano e avremmo dovuto seguire altri step per modificarlo in base alle nostre esigenze.

APPLICAZIONE

Tramite la CamLibrary qualsiasi applicazione può prendere il controllo della webcam, inizializzarla, calibrarla in modo da determinare il modello del target e individuare il punto del laser in coordinate relative al centro del modello. Ad esempio, da un'applicazione .Net possiamo interfacciarci con la CamLibrary nel modo seguente. Per prima cosa bisogna inserire fra la lista dei namespace referenziati bisogna la seguente riga:

```
using System.Runtime.InteropServices;
```

Poi copiamo la DLL creata precedentemente nella stessa cartella dell'eseguibile in modo che la possa vedere; infine, dopo la definizione della classe introduciamo le varie chiamate ai metodi esportati dalla CamLibrary

```
[DllImport("CamLibrary.dll")]
public static extern bool InitCamera();
```

utilizziamo DllImport per indicare al compilatore che stiamo importando una funzione dalla libreria CamLibrary.dll, di seguito troviamo il prototipo della funziona esportata. A questo punto abbiamo proprio tutto: la libreria di interfaccia e gli strumenti per utilizzarla. Per realizzare il gioco basta ricordarsi di:

- inizializzare la webcam
- effettuare delle istantanee
- calibrare il dispositivo fino a trovare un modello ricostruito abbastanza soddisfacente del target, chiamando ripetutamente Find_Circles (dopo aver recuperato l'istantanea)
- una volta calibrata la videocamera digitale si procede con l'individuazione del dot richiamando il metodo Find_Shot dopo aver impostato un adeguato lasso di tempo,

TECNICA

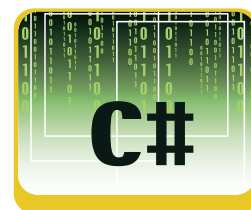
La costruzione della CamLibrary ci ha semplificato di molto lo sviluppo dell'applicazione in sé. Infatti, il tutto si riduce ad istanziare un timer che ad intervalli di tempo prefissati esegue nel suo gestore del tick una serie di controlli sull'istantanea prodotta. Ad esempio, il modulo di calibrazione è subito realizzato nel modo seguente:

```
private void timerCalibra_Tick(object sender,
                                        EventArgs e)
{bool trovatoC=false;
 timerCalibra.Enabled=false;
 if (GrabFrame()){
  System.IntPtr intPtr=RetrieveFrame();
  Bitmap bitmap=Bitmap.FromHbitmap(intPtr);
  video.Immagine=bitmap;
  if (trovatoC=Find_Circles()){
   centro[0]=Centro_X();
   centro[1]=Centro_Y();
   raggio=Centro_R();
   video.Centro=new Point(centro[0],centro[1]);
   video.Raggio=raggio;
   game.Raggio=raggio;
   game.Centro=new Point(centro[0],centro[1]);
   btnGioca.ImageIndex=3;
   toolTip1.SetToolTip(btnGioca, "Nuova partita");}
  }
  timerCalibra.Enabled=true;
}
```

Si comincia "scattando" l'istantanea richiamando GrabFrame() e recuperando la bitmap in memoria per poterla visualizzare, dopo di che si richiama il metodo Find_Circles() su questa immagine in memoria e se la procedura va a buon fine si recuperano le coordinare X e Y del centro e il raggio del cerchio base.
Il modulo per l'individuazione del punto del laser è molto simile

```
private void timerGioca_Tick(object sender, EventArgs e)
{
  bool trovatoS=false;
  timerGioca.Enabled=false;
  if (GrabFrame())
  {
   System.IntPtr intPtr=RetrieveFrame();
   Bitmap bitmap=Bitmap.FromHbitmap(intPtr);
   video.Immagine=bitmap;
   Find_Shot();
   if((new TimeSpan(DateTime.Now.Ticks-
                              lastShot)).Milliseconds>700)
   {trovatoS=Shot_X()!=0 && Shot_Y()!=0;
    game.Abilita();
    if(trovatoS)
    {
          shot[0]=Shot_X();
          shot[1]=Shot_Y();
          lastShot=DateTime.Now.Ticks;
          game.Shot=new Point(Shot_X(),Shot_Y());
    }
   }
  }
  timerGioca.Enabled=true;
}
```

Le coordinare sono in sistema relativo al centro del modello ricostruito, quindi tengono conto di quanto "distano" dal centro stesso.

Ing. Antonino Panella

NOTE

Panella è un ingegnere informatico impegnato da anni nello sviluppo di tecniche basate sulla Computer Vision, anche in ambito mobile su dispositivi Pocket Pc. Per informazioni rivolgersi alla sua mailbox: antonino.panella@google.com

APPLICAZIONE

Tramite la CamLibrary qualsiasi applicazione può prendere il controllo della webcam, inizializzarla, calibrarla in modo da determinare il modello del target e individuare il punto del laser in coordinate relative al centro del modello. Ad esempio, da un'applicazione .Net possiamo interfacciarci con la CamLibrary nel modo seguente. Per prima cosa bisogna inserire fra la lista dei namespace referenziati bisogna la seguente riga:

```
using System.Runtime.InteropServices;
```

Poi copiamo la DLL creata precedentemente nella stessa cartella dell'eseguibile in modo che la possa vedere; infine, dopo la definizione della classe introduciamo le varie chiamate ai metodi esportati dalla CamLibrary

```
[DllImport("CamLibrary.dll")]
public static extern bool InitCamera();
```

utilizziamo DllImport per indicare al compilatore che stiamo importando una funzione dalla libreria CamLibrary.dll, di seguito troviamo il prototipo della funziona esportata. A questo punto abbiamo proprio tutto: la libreria di interfaccia e gli strumenti per utilizzarla. Per realizzare il gioco basta ricordarsi di:

- inizializzare la webcam
- effettuare delle istantanee
- calibrare il dispositivo fino a trovare un modello ricostruito abbastanza soddisfacente del target, chiamando ripetutamente Find_Circles (dopo aver recuperato l'istantanea)
- una volta calibrata la videocamera digitale si procede con l'individuazione del dot richiamando il metodo Find_Shot dopo aver impostato un adeguato lasso di tempo,

TECNICA

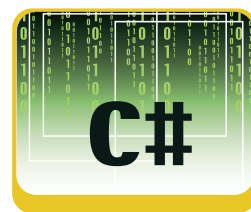
La costruzione della CamLibrary ci ha semplificato di molto lo sviluppo dell'applicazione in sé. Infatti, il tutto si riduce ad istanziare un timer che ad intervalli di tempo prefissati esegue nel suo gestore del tick una serie di controlli sull'istantanea prodotta. Ad esempio, il modulo di calibrazione è subito realizzato nel modo seguente:

```
private void timerCalibra_Tick(object sender,
                                    EventArgs e)
{bool trovatoC=false;
 timerCalibra.Enabled=false;
 if (GrabFrame()){
  System.IntPtr intPtr=RetrieveFrame();
  Bitmap bitmap=Bitmap.FromHbitmap(intPtr);
  video.Immagine=bitmap;
  if (trovatoC=Find_Circles()){
   centro[0]=Centro_X();
   centro[1]=Centro_Y();
   raggio=Centro_R();
   video.Centro=new Point(centro[0],centro[1]);
   video.Raggio=raggio;
   game.Raggio=raggio;
   game.Centro=new Point(centro[0],centro[1]);
   btnGioca.ImageIndex=3;
   toolTip1.SetToolTip(btnGioca, "Nuova partita");}
  }
  timerCalibra.Enabled=true;
}
```

Si comincia "scattando" l'istantanea richiamando GrabFrame() e recuperando la bitmap in memoria per poterla visualizzare, dopo di che si richiama il metodo Find_Circles() su questa immagine in memoria e se la procedura va a buon fine si recuperano le coordinare X e Y del centro e il raggio del cerchio base.
Il modulo per l'individuazione del punto del laser è molto simile

```
private void timerGioca_Tick(object sender, EventArgs e)
{
  bool trovatoS=false;
  timerGioca.Enabled=false;
  if (GrabFrame())
  {
   System.IntPtr intPtr=RetrieveFrame();
   Bitmap bitmap=Bitmap.FromHbitmap(intPtr);
   video.Immagine=bitmap;
   Find_Shot();
   if((new TimeSpan(DateTime.Now.Ticks-
                         lastShot)).Milliseconds>700)
   {trovatoS=Shot_X()!=0 && Shot_Y()!=0;
    game.Abilita();
    if(trovatoS)
    {
          shot[0]=Shot_X();
          shot[1]=Shot_Y();
          lastShot=DateTime.Now.Ticks;
          game.Shot=new Point(Shot_X(),Shot_Y());
    }
   }
  }
  timerGioca.Enabled=true;
}
```

Le coordinare sono in sistema relativo al centro del modello ricostruito, quindi tengono conto di quanto "distano" dal centro stesso.

Antonino Panella

NOTE

Panella è un ingegnere informatico impegnato da anni nello sviluppo di tecniche basate sulla Computer Vision, anche in ambito mobile su dispositivi Pocket Pc. Per informazioni rivolgersi alla sua mailbox: antonino.panella@google.com

VISUALIZZARE IMMAGINI DA UNA WEBCAM

LE AVICAP WINDOW CLASS, MESSE A DISPOSIZIONE DAL SISTEMA OPERATIVO WINDOWS PERMETTONO DI GESTIRE, FACILMENTE, I PROCESSI DI ACQUISIZIONE PROVENIENTI DA QUALSIASI DISPOSITIVO VIDEO

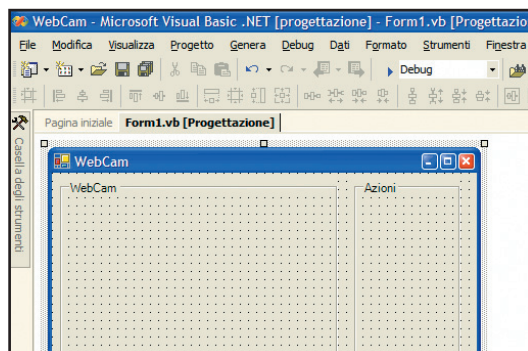
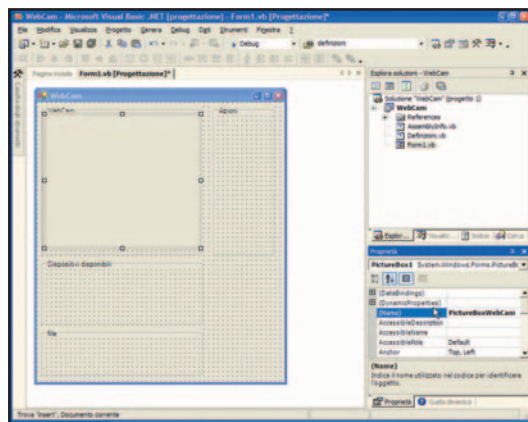
Attualmente le webcam sono periferiche di uso molto comune per l'utilizzo in chat o in videoconferenza, anche per il costo ridotto alla portata di tutti. In questo articolo vedremo come acquisire immagini tramite una webcam in una applicazione .Net per mezzo delle classi *AVICap*.
La libreria *AVIcap window class (avicap32.dll)* raggruppa, e permette di gestire, le funzionalità multimediali che l'ambiente Windows mette a disposizione per interfacciarsi con qualsiasi dispositivo audio-visivo, senza doverci preoccupare dei dispositivi stessi
La potenzialità maggiore della libreria è l'interoperabilità con qualunque tipo di dispositivo di acquisizione installato nel sistema.

IL NAMESPACE SYSTEM.RUNTIME .INTEROPSERVICES

Per accedere al flusso di dati inviati dalla nostra webcam dovremo utilizzare due DLL di sistema, l'*Avicap32.dll* e la *User32.dll,* e per farlo è necessario creare delle funzioni che faranno da puntatori alle funzioni interne delle nostre dll.
Il namespace *System.Runtime.InteropServices* mette a disposizione metodi ed attributi che ci permetteranno di lavorare con COM e servizi di chiamata al sistema operativo.
In particolare, in Visual Basic la conversione degli argomenti in tipi di dati compatibili, un processo definito *marshalling*, viene eseguita automaticamente. Per controllare in modo esplicito il *marshalling* degli argomenti è possibile utilizzare l'attributo *MarshalAs*.
In tutti gli esempi di codice, per evitare di scrivere ogni volta il nome completo della libreria, diamo per scontato l'inserimento della seguente istruzione *Imports:*

```
Imports System
```

```
Imports System.Runtime.InteropServices
```

1 Per iniziare, creiamo un nuovo progetto *Windows Applications* e sulla finestra *Form1* disegniamo l'interfaccia utente, in particolare piazziamo un po' di controlli contenitore *GroupBox* per ingentilire la finestra e per raggruppare i controlli.

2 Selezioniamo un controllo *PictureBox* dalla casella degli strumenti, e disegniamolo sulla form. Dalla finestra delle proprietà cambiamo il nome in *PictureBoxWebCam.* In *PictureBoxWebCam* visualizzeremo le immagini provenienti dalla webcam.

3 Selezioniamo un controllo *ListBox* dalla casella degli strumenti, e disegniamolo

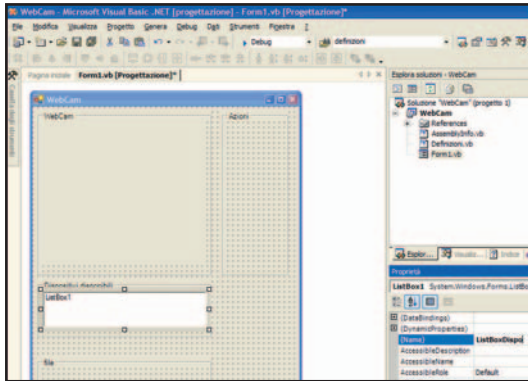
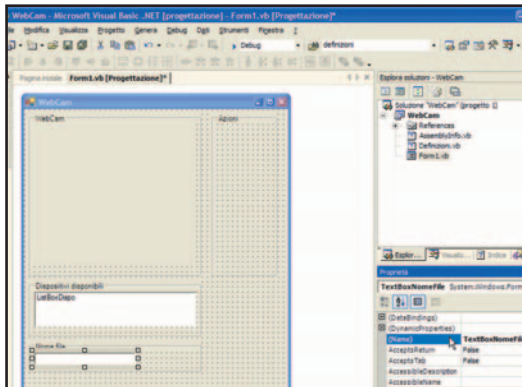
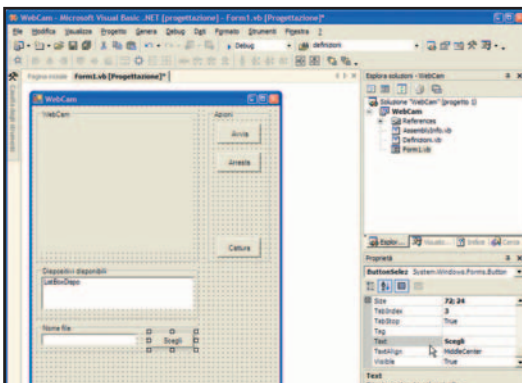
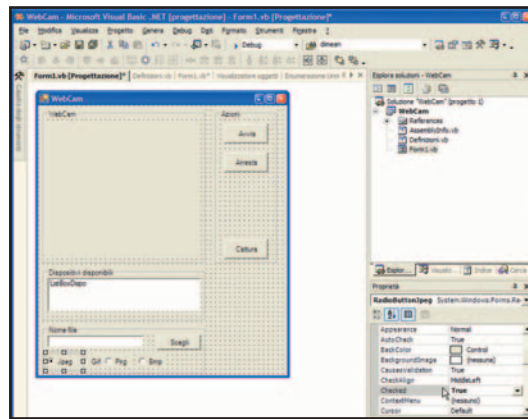
REQUISITI

Conoscenze richieste
NET Framework, Sviluppo Windows

Software
Sistema operativo:Windows 2000/XP. Visual Basic .NET 2003.

Impegno

Tempo di realizzazione

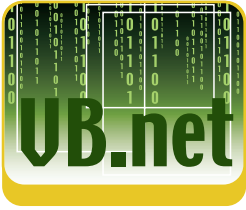
sulla form. Dalla finestra delle proprietà cambiamo il nome in *ListBoxDispo*. In *ListBoxDispo* visualizzeremo i nomi dei dispositivi di acquisizione video presenti nel sistema.

4 Selezioniamo un controllo *TextBox* dalla casella degli strumenti, e disegniamolo sulla form. Dalla finestra delle proprietà cambiamo il nome in *TextBoxNomeFile*. In *TextBoxNomeFile* visualizzeremo il nome del file in cui verrà memorizzata l'immagine catturata dalla webcam.

5 Selezioniamo quattro controlli *CommandButton* dalla casella degli strumenti, e disegniamoli sulla form. Dalla finestra delle proprietà cambiamo i nomi in *ButtonAvvia, ButtonArresta, ButtonCattura, ButtonScegli*. *ButtonScegli* ci permetterà di scegliere il nome del file da una finestra di dialogo, gli altri *CommandButton* ci permetteranno di gestire la webcam.

6 Selezioniamo quattro controlli *RadioButton* dalla casella degli strumenti, e disegniamoli sulla form. Dalla finestra delle proprietà cambiamo il nome in *RadioButtonJpeg, RadioButtonGif, RadioButtonPng, RadioButtonBmp*. Selezionando uno dei quattro *RadioButton*, si può scegliere il formato dell'immagine catturata dalla webcam

ELENCO DEI DISPOSITIVI INSTALLATI

Prima di compiere qualsiasi operazione con immagini provenienti da una webcam, dobbiamo, ovviamente, assicurarci di avere una webcam e che sia correttamente installata. Per questo motivo, dovremo controllare l'esistenza di almeno un dispositivo di acquisizione video.

Nel nostro progetto aggiungiamo un modulo *Definizioni*, nel quale inseriremo: le dichiarazioni delle API, le dichiarazioni delle costanti da utilizzare con *AVICap window class* e le dichiarazioni delle routine di interfacciamento con il dispositivo di acquisizione.

PRIMA DI VB.NET?

Nelle versioni precedenti di Visual Basic era possibile dichiarare parametri As Any, che indicavano la possibilità di utilizzare qualsiasi tipo di dati. In Visual Basic .NET è necessario utilizzare un tipo di dati specifico in tutte le istruzioni di dichiarazione. Nel nostro caso possiamo utilizzare l'attributo MarshalAs con il compito di indicare che il tipo di dati sottostante il parametro lParam deve essere sottoposta a marshalling come struttura di tipo. Per il parametro lParam della SendMessage utilizziamo il membro AsAny dell'enumerazione UnmanagedType. AsAny rappresenta un tipo dinamico che determina il tipo di un oggetto in fase di esecuzione ed effettua il marshalling dell'oggetto come suddetto tipo.

È possibile utilizzare la parola chiave *Alias* per indicare che il nome della routine da chiamare è diverso nella DLL.

Nel modulo *Definizioni* definiamo la funzione *capGetDriverDescription* che fa riferimento a una funzione esistente in *avicap32.dll.* Questa funzione restituisce *True* se esiste un dispositivo di acquisizione, o *False* nel caso contrario

```
Public Declare Function capGetDriverDescription Lib
        "avicap32.dll" Alias "capGetDriverDescriptionA"
    (ByVal wDriverIndex As Short, ByVal lpszName As
    String, ByVal cbName As Integer, ByVal lpszVer As
            String, ByVal cbVer As Integer) As Boolean
```

I parametri che si devono passare alla funzione, con valore non nullo, sono:

- **wDriverIndex.** Indice del dispositivo di acquisizione. Può variare da 0 a 9.
- **lpszName.** Punta al buffer che contiene il nome del dispositivo di acquisizione.
- **cbName.** Indica la lunghezza, in bytes, del buffer puntato da *lpszName.*
- **lpszVer.** Punta al buffer che contiene la versione del dispositivo di acquisizione.
- **cbVer.** Indica la lunghezza, in byte, del buffer puntato da *lpszVer.*

La lista dei dispositivi disponibili, dovrà essere visualizzata all'apertura dell'applicazione, per questo scriviamo il codice necessario nell'evento *Load* di *Form1*

```
Private Sub Form1_Load(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles MyBase.Load

End Sub
```

Definiamo due variabili *(NomeDispositivo, Versione)* in cui memorizzare il nome e la versione del dispositivo di acquisizione. È necessario inizializzare queste variabili perché in caso contrario anche dopo il loro utilizzo, risulteranno sempre *null.*
Definiamo inoltre una variabile di controllo ed una variabile indice

```
    Dim NomeDispositivo As String = Space(100)
    Dim Versione As String = Space(100)
    Dim Dispositivo As Boolean
    Dim i As Integer
```

Scriviamo un ciclo in cui elenchiamo tutti i dispositivi disponibili e li aggiungiamo in *ListBoxDispo*

```
    Do
        Dispositivo = capGetDriverDescription(
            i, NomeDispositivo, 100, Versione, 100)
        If Dispositivo Then ListBoxDispo.Items.Add(
                                   NomeDispositivo)
        i += 1
    Loop Until Dispositivo = False
```

Per effetto di queste istruzioni possiamo vedere elencati i dispositivi installati nel sistema, nel caso del sistema di test è presente una *WebCam Creative* come si può vedere in figura, ma il risultato si ottiene per qualsiasi altra webcam.

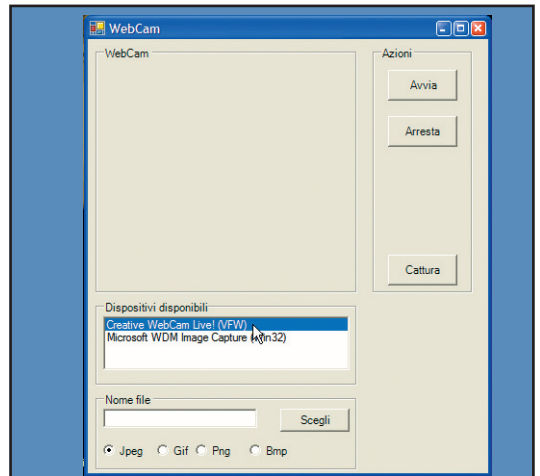



Fig. 1: ***La lista dei dispositivi installati nel sistema***

A questo punto l'utente potrà selezionare il dispositivo di acquisizione da utilizzare, semplicemente cliccando con il mouse nella lista. Il codice necessario dovrà essere scritto, quindi, nell'evento *SelectedIndexChanged* della *ListBox.*

```
Private Sub ListBoxDispo_SelectedIndexChanged(
            ByVal sender As System.Object, ByVal e As
                       System.EventArgs) Handles
                 ListBoxDispo.SelectedIndexChanged
    idDispositivo = ListBoxDispo.SelectedIndex
End Sub
```

Dove *idDispositivo* è una variabile definita in form1 in cui memorizzare il numero identificativo del dispositivo di acquisizione selezionato

```
Dim idDispositivo As Integer
```

MODULO DEFINIZIONI

Ora che abbiamo selezionato il nostro dispositivo di acquisizione, siamo pronti per realiz-

zare un piccolo riquadro di preview, che ci farà vedere in tempo reale le immagini che la nostra webcam invia al sistema. Per far ciò dobbiamo utilizzare, ancora, delle funzioni messe a disposizione dalle DLL *avicap32* e *user32*, per questo motivo sempre nel modulo *Definizioni* scriviamo:

```
Public Declare Function capCreateCaptureWindow Lib
    "avicap32.dll" Alias "capCreateCaptureWindowA"
 (ByVal lpszWindowName As String, ByVal dwStyle As
    Integer, ByVal x As Integer, ByVal y As Integer,
                                              ByVal
    nWidth As Integer, ByVal nHeight As Short, ByVal
    hWnd As Integer, ByVal nID As Integer) As Integer
```

La funzione *capCreateCaptureWindow* permette l'utilizzo di una superficie per la visualizzazione e la cattura di un segnale video esterno. I parametri che si devono passare alla funzione, con valore non nullo, sono:

- **lpszWindowName.** Indica il nome della superficie di acquisizione
- **dwstyle.** Indica lo stile della superficie di acquisizione
- **X, Y, nWidth, nHeight.** Indicano il posizionamento e le dimensioni della superficie di acquisizione
- **hWnd.** Indica l'handle del controllo contenitore della superficie di acquisizione.

La funzione *capCreateCaptureWindow* restituisce un valore di tipo *Long*, che rappresenta il valore di handle da utilizzare nella finestra di acquisizione per la connessione con il driver di cattura.
La connessione al driver di acquisizione viene effettuata tramite la funzione *SendMessage* di *user32.dll* che permette di inviare messaggi alla *avicap32*, dopo aver impostato il flusso di dati

```
Public Declare Function SendMessage Lib "user32"
 Alias "SendMessageA" (ByVal hwnd As Integer, ByVal
    wMsg As Integer, ByVal wParam As Integer,
  <MarshalAs(UnmanagedType.AsAny)> ByVal lParam
    As Object) As Integer
```

Infine dobbiamo definire la funzione *SetWindowPos* che verrà utilizzata per ridimensionare la finestra del flusso di dati.

```
Public Declare Function SetWindowPos Lib "user32"
 Alias "SetWindowPos" (ByVal hwnd As Integer, ByVal
    hWndInsertAfter As Integer, ByVal x As Integer,
 ByVal y As Integer, ByVal cx As Integer, ByVal cy As
    Integer, ByVal wFlags As Integer) As Integer
```

IL RIQUADRO DI ANTEPRIMA

Torniamo nella finestra del codice di form1 e definiamo la variabile in cui memorizzare l'handle da utilizzare nella finestra di acquisizione, per la connessione con il driver di cattura

```
Dim hHwnd As Integer
```

Il codice necessario a visualizzare le immagini provenienti dalla webcam, dovrà essere scritto nell'evento *click* di *ButtonAvvia:*

```
Private Sub ButtonAvvia_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles ButtonAvvia.Click

End Sub
```

All'interno di *ButtonAvvia_Click* definiamo due variabili in cui memorizzare l'altezza e la larghezza del *PictureBox (PictureBoxWebCam)* in cui visualizzeremo le immagini

```
    Dim iHeight As Integer =
        PictureBoxWebCam.Height
    Dim iWidth As Integer =
        PictureBoxWebCam.Width
```

Utilizziamo la funzione *capCreateCaptureWindow* per indicare al sistema che il flusso di dati dovrà essere inviato al nostro *PictureBox*, passando come parametro il valore *PictureBoxWebCam.Handle.ToInt32* che identifica univocamente il controllo.
Oltre a questo parametro passiamo: la dimensione del frame di cattura (640, 480), e le costanti *WS_VISIBLE* e *WS_CHILD* (vedi box)

```
    hHwnd = capCreateCaptureWindow(
        "Anteprima", WS_VISIBLE Or WS_CHILD,
        0, 0, 640, _480,
        PictureBoxWebCam.Handle.ToInt32, 0)
```

A questo punto dobbiamo collegarci alla webcam richiamando la funzione *SendMessage*, passando come parametri: l'handle del PictureBox, la costante *WM_CAP_DRIVER_CONNECT* e l'identificativo del dispositivo selezionato.

GRABBING
Il termine *grabbing* identifica il procedimento di cattura di un singolo fotogramma dal flusso video riprodotto in una superficie di acquisizione.

STREAMING
Il termine *streaming* identifica, il procedimento di registrazione di una sequenza (*stream*) dal flusso video riprodotto in una superficie di acquisizione.

```
        If SendMessage(hHwnd,
                     WM_CAP_DRIVER_CONNECT,
                              idDispositivo, 0) Then
```

È necessario il test sul valore restituito dalla funzione poiché si può verificare un fallimento *(risultato=0)* in uno dei tre casi seguenti:

- Nel sistema non sono installati dispositivi di acquisizione.
- il dispositivo è installato, ma non correttamente.
- il dispositivo è installato correttamente ma è già attivo un altro applicativo connesso con il driver. Ad ogni driver infatti può essere connessa una sola superficie di visualizzazione, e di cattura, alla volta.

Ora che siamo connessi con la nostra webcam, dobbiamo impostare le dimensioni del flusso di immagini in base alle dimensioni del nostro controllo di anteprima, impostare i frame di acquisizione e far partire il processo di acquisizione:

```
            'Impostiamo la scala dell'anteprima
            SendMessage(hHwnd, WM_CAP_SET_SCALE,
                                            True, 0)

            'Impostiamo il rate in millisecondi
            SendMessage(hHwnd,
                     WM_CAP_SET_PREVIEWRATE, 66, 0)

            'Facciamo partire l'anteprima
            SendMessage(hHwnd,
                         WM_CAP_SET_PREVIEW, True, 0)

            'Scaliamo le immagini alle dimensioni del
                                    nostro PictureBox
            SetWindowPos(hHwnd, HWND_BOTTOM, 0,
                   0, iWidth, iHeight, _SWP_NOMOVE Or
                                        SWP_NOZORDER)
```

Infine nel ramo *Else*, in cui si entra nel caso di dispositivo non presente, chiudiamo la superficie di acquisizione utilizzando la funzione *DestroyWindow*, presente nella *user32dll*. *DestroyWindow* dovrà essere definita sempre nel modulo definizioni con la seguente sintassi:

```
Public Declare Function DestroyWindow Lib "user32"
                    (ByVal hndw As Integer) As Boolean
```

La funzione *DestroyWindow* aspetta come unico parametro l'handle della nostra superficie di acquisizione, per questo possiamo scrivere:

```
Else
            DestroyWindow(hHwnd)
    End If
```

Come risultato del codice appena scritto, nel momento in cui premiamo il pulsante *ButtonAvvia* vedremo comparire all'interno del *pictureBox*, in tempo reale, le immagini provenienti dalla nostra Webcam.

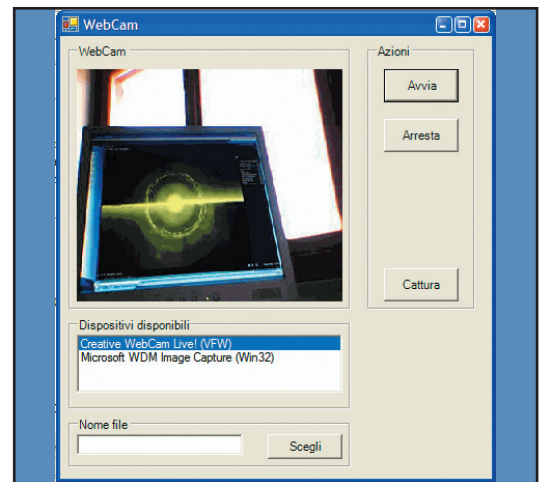



Fig. 2: ***Le immagini provenienti dalla webcam***

PULSANTE DI ARRESTO

Per terminare le operazioni di acquisizione, l'utente dovrà premere il pulsante *ButtonArresta*, per questo dobbiamo scrivere il codice necessario nell'evento *Click*.

```
Private Sub ButtonArresta_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                          Handles ButtonArresta.Click

End Sub
```

Le operazioni da compiere saranno: disconnettere la periferica, chiudere la superficie di acquisizione. Per questo utilizziamo sempre la funzione *SendMessage*, passando come parametri: l'handle del PictureBox, la costante *WM_CAP_DRIVER_DISCONNECT* e l'identificativo del dispositivo selezionato. Infine utilizziamo la funzione *DestroyWindow* vista in precedenza

```
        SendMessage(hHwnd,
                    WM_CAP_DRIVER_DISCONNECT,
                                 idDispositivo, 0)
        DestroyWindow(hHwnd)
```

NOTA

La classe *Bitmap* incapsula una bitmap GDI+ costituita dai pixel di un'immagine e dai relativi attributi. *Bitmap* è un oggetto utilizzato per operare con le immagini definite dai dati pixel. Il metodo *Save* della classe *Bitmap* salva l'immagine nel file e nel formato specificati.

SALVARE L'IMMAGINE IN UN FILE

Dopo aver attivato la nostra WebCam possiamo catturare un singolo frame e salvarlo in un file immagine. La prima operazione da compiere è quella di scegliere il nome del file in cui conservare l'immagine, per questo scriviamo il codice necessario nell'evento *Click* del pulsante *ButtonSelez*

```
Private Sub ButtonSelez_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        Handles ButtonSelez.Click
    SaveFileDialog1.ShowDialog()
    TextBoxNomeFile.Text = SaveFileDialog1.FileName
End Sub
```

Molto semplicemente abbiamo mostrato la finestra di dialogo *"Salva"* in cui abbiamo selezionato il path ed il nome del file, ed abbiamo mostrato la scelta in *TextBoxNomeFile.*
Una volta scelto il nome del file, possiamo cliccare sul pulsante *"Cattura"* per avviare il processo di salvataggio del frame catturato.
Scriviamo il codice necessario nell'evento *Click* di *ButtonCattura*

```
Private Sub ButtonCattura_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                        Handles ButtonCattura.Click
End Sub
```

Dichiariamo una variabile *NomeFile* che dovrà contenere il nome del file selezionato. La libreria *avicap32.dll* salva esclusivamente in formato bmp, per questo se l'utente seleziona un formato diverso, dovremo salvare il risultato in un file temporaneo che sarà oggetto di una successiva decodifica.

```
    Dim NomeFile As String
    If RadioButtonBmp.Checked Then
        NomeFile = TextBoxNomeFile.Text & ".bmp"
    Else
        NomeFile = TextBoxNomeFile.Text &
                                "temp.bmp"
    End If
```

Per salvare il frame in un file, è sufficiente eseguire una chiamata alla funzione *SendMessage* passando come parametro la costante per il salvataggio ed il percorso del file

```
    SendMessage(hHwnd, WM_CAP_FILE_SAVEDIB,
                                0&, NomeFile)
```

Se l'utente ha selezionato un formato diverso dal formato *Bmp*, dobbiamo avviare la conversione e cancellare il file temporaneo

```
    If RadioButtonBmp.Checked = False Then
        ConvertiDaBMP(NomeFile)
        System.IO.File.Delete(NomeFile)
    End If
```

La procedura *ConvertiDaBMP* riceve il nome del file temporaneo da convertire, e genera un nuovo file nel formato selezionato. Per ottenere il risultato voluto, utilizziamo la classe *Bitmap* che abbiamo ampiamente descritto in un precedente articolo sul GDI+ (per utilizzare la classe *Bitmap* dobbiamo scrivere una istruzione *Import* in cima al codice di *Form1: Imports System.Drawing.Imaging*)

```
Private Sub ConvertiDaBMP(ByVal NomeFile As String)
    Try
        Dim objBmp As New Bitmap(NomeFile)
        If RadioButtonJpeg.Checked Then
            objBmp.Save(TextBoxNomeFile.Text &
                        ".jpg", ImageFormat.Jpeg)
        End If
        If RadioButtonGif.Checked Then
            objBmp.Save(TextBoxNomeFile.Text &
                        ".gif", ImageFormat.Gif)
        End If
        If RadioButtonPng.Checked Then
            objBmp.Save(TextBoxNomeFile.Text &
                        ".Png", ImageFormat.Png)
        End If
        objBmp.Dispose()
    Catch
    End Try
End Sub
```

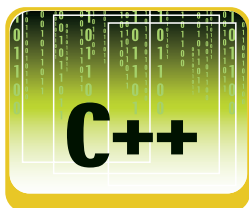
Luigi Buono

L'ELENCO DI COSTANTI NECESSARIE PER DIALOGARE CON LA DLL AVICAP32

```
Public Const WM_CAP As Short =
                        &H400S
Public Const WM_USER As Long =
                        &H400

Public Const
    WM_CAP_DRIVER_CONNECT As
        Integer = WM_CAP + 10
Public Const
    WM_CAP_DRIVER_DISCONNECT
        As Integer = WM_CAP + 11
Public Const WM_CAP_EDIT_COPY
        As Integer = WM_CAP + 30
Public Const
    WM_CAP_FILE_SAVEDIB As Long
        = WM_USER + 25
Public Const WM_CAP_FILE_SAVEAS
        As Long = WM_USER + 23
Public Const
    WM_CAP_SET_PREVIEW As
        Integer = WM_CAP + 50
Public Const
    WM_CAP_SET_PREVIEWRATE As
        Integer = WM_CAP + 52
Public Const WM_CAP_SET_SCALE
        As Integer = WM_CAP + 53
Public Const WS_CHILD As Integer
        = &H40000000
Public Const WS_VISIBLE As
        Integer = &H10000000
Public Const SWP_NOMOVE As
        Short = &H2S
Public Const SWP_NOSIZE As Short
        = 1
Public Const SWP_NOZORDER As
        Short = &H4S
Public Const HWND_BOTTOM As
        Short = 1
```

SKELETAL ANIMATION

ANIMADEAD È UNA LIBRERIA PROGETTATA E REALIZZATA PER CONSENTIRE A QUALUNQUE APPLICAZIONE DI ESEGUIRE ANIMAZIONI DI FIGURE TRIDIMENSIONALI CON UNA TECNICA CHE LE RENDE PARTICOLARMENTE REALISTICHE. VEDIAMO QUALE

Molti di voi saranno sicuramente rimasti affascinati dagli effetti visivi di giochi come *Unreal Tournament 2004* della Epic Games o come *Splinter Cell* della Ubi Soft. In questi giochi i vari personaggi che si aggirano sullo schermo si muovono, camminano e combattono in un modo molto realistico. Certamente tanto è dovuto alla capacità di calcolo delle console e dei PC di ultima generazione, ma questo realismo non si sarebbe mai potuto ottenere, neppure con il più sofisticato sistema di intelligenza artificiale, se il mondo della programmazione "ludica" non avesse tratto ispirazione dall'anatomia. Infatti, il nostro corpo si muove grazie a quello che lo sostiene e gli dona rigidità e cioè lo scheletro. Il principio di base è lo stesso: i personaggi di questi giochi sono stati "costruiti" in grafica dotandoli di uno scheletro (spesso chiamato sostegno o armatura interna della figura) e legando il corpo esterno a questo scheletro. In questo modo si ottiene un pupazzo virtuale che si può muovere e sistemare come si farebbe con una bambola vera. Ciò è molto utile nei casi in cui il movimento del modello deve essere comandato dall'utente, come appunto in molti giochi per computer e console, ma che può trovare riscontro anche in altri settori.
Tramite una telecamera è possibile, ad esempio, studiare il movimento di una persona ed utilizzando oppurtuni marcatori visivi si costruisce un modello tridimensionale del suo scheletro. In questo modo, si ottiene una figura che può assumere diverse pose (preimpostate) e l'effetto del movimento lo si crea attraverso l'assegnazione di una delle pose in un certo intervallo di tempo.

COSTRUIRE IL MODELLO

La prima cosa da fare quindi è avere ben in mente la figura finale di cui si vuole ottenere il modello 3D, disegnarlo con un buon programma di grafica, come 3D Studio Max e Maya. per poi "introdurvi" lo scheletro. Consideriamo il seguente esempio. Prendiamo una figura semplice che ricorda un uomo stilizzato, senza mani e piedi, una forma tipica degli omini di pan di zenzero tanto diffusi nella cultura anglossassone.

AGGIUNGERE LO SCHELETRO

Il passo successivo sarà quello di dargli uno armatura (lo scheletro): lo scopo dell'armatura sarà quello di vincolare i movimenti in modo da irrigidire la struttura ed impedire di assumere pose innaturali (ad esempio un ginocchio che si piega all'indietro). La semplicità del personaggio è tale da non richiedere un'armatura troppo complessa, basteranno quattro arti (due braccia e due gambe) ed un paio di giunture per gli arti inferiori, solo le ginocchia e non i gomiti. La presenza di collo, mani e piedi avrebbe complicato la struttura, sempre però relizzabile con un pò di pazienza. A questo punto abbiamo già il modello, ma ancora corpo e scheletro sono slegati: bisogna infatti far aderire il rivestimento all'armatura e vincolarli in modo che che una deformazione dello scheletro provochi una deformazione del corpo. Ciò è ottenibile assegnando i vertici alle ossa in modo che ogni porzione di corpo abbia un suo osso e che ciascun osso sia legato ad un altro in modo da vincolare movimenti della figura.

DEV-C++

Bloodshed Dev-C++ è un ambiente di sviluppo integrato scritto in Delphi completamente gratuito (Open Source), con tutte le funzionalità richieste, per la programmazione in C e C++ dove il compilatore usato è Mingw che fa parte della GNU Compiler Collection. Gira su sistemi Win32, a partire da Windows 95 a XP, sia come console o con interfaccia grafica. Dev-C++ può anche essere usato congiuntamente a Cygwin o a qualunque altro compilatore basato GCC.

IN POSA!

Adesso il personaggio è veramente finito! All'interno del programma di grafica possiamo giocare con questo modello come se fosse una bambola, pronta ad assumere tutte le pose che ci vengono in mente, sempre tenendo presente il vincolo dello scheletro. Si può costruire una posa impostando la posizione e la rotazione di ciascun osso: dato che esiste un legame con il rivestimento, in base allo scheletro il pro-

gramma di rendering aggiornerà l'esterno in modo adattarlo alla posa scelta. L'animazione sarà quindi composta da singole pose. Utilizzando la stessa armatura è possibile costruire diverse animazioni in modo da soddisfare tutte le esigenze "sceniche" che si dovessere incontrare: ad esempio, si potrebbe individuare anche un movimento delle braccia o di rotazione del busto oltre naturalemente alla possibilità di far camminare il personaggio.

SKELETAL ANIMATION

Da una parte abbiamo quindi le meshes, tutte le forme geometriche e il legame delle strutture che costituiscono il look complessivo (lo scheletro della figura, e naturalmente il legame con il rivestimento), mentre dall'altra ci sono le animations cioè tutti i possibili movimenti previsti (l'insieme delle pose per l'oggetto ad ogni intervallo di tempo). Mantenere separate le meshes dalle animations può tornare molto ultile quando si vuole avere un modello sempre presente da adattare a diverse situazioni, un pò come giocare apunto con una bambola. Come è possibile ottenere quel livello di realismo visto nei giochi di cui abbiamo parlato all'inizio? Beh, la necessità aguzza l'ingegno, basta partire da un'osservazione. Una volta realizzato il modello tramite Maya, oltre che renderizzare la figura questo programma può svolgere anche il ruolo di "burattinaio", cioè lo possiamo usare per manipolare la figura per farla assumere le pose che vogliamo. Allora perché non posizionarsi ad un livello intermedio tra quello tutto grafico e quello tutto runtime? Tramite Maya renderizziamo il modello e si memorizzano i suoi possibili movimenti in pose senza però realizzare il filmato. Il grosso lavoro di grafica è già bello che fatto, dato che il legame corpo-scheletro è pronto e le pose sono state riprese. A questo punto il modello può essere importato da opportuni software in modo da agirci in runtime lavorando sulle diverse pose.

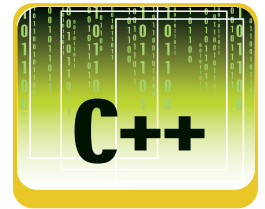
ANIMADEAD

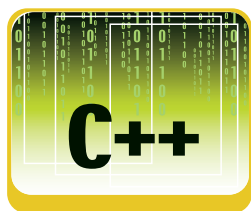
Animadead è una libreria che consente di utilizzare all'interno di un'applicazione su modello 3D proprio a questo livello intermedio. È scritta interamente in C++ ed è indipendente dalla piattaforma, per cui si adatta molto bene a diverse piattaforme. Inoltre è un progetto Open Source, distribuita sotto licenza GNU LGPL (versione 2.1), quindi completamente gratuito e può trovare impiego anche in ambito commerciale, distribuendo oltre all'applicazione che contiene la libreria anche il file di licenza. Il cuore di questa libreria è l'OpenGL e la *Simple DirectMedia Layer* (vedere il relativo box).
La distribuzione della versione 2.0 di Animadead (il file *animadead-2.0.tar* e lo si può trovare all'indirizzo *http://animadead.sourceforge.net*) contiene i sorgenti C++ della libreria e di un esempio di utilizzo, un plug-in di esportazione per Maya e due rilasci per aiutare la compilazione in ambiente Win32, uno per Visual Studio .Net 2003 e l'altro per Dev-C++.

COMPILARE SDL

Come già detto, la distribuzione non comprende l'SDL; faremo uso della versione dei sorgenti della versione 1.2.9, necessari per ottenere anche nel caso dell'SDL una libreria di runtime. Estraendo il contenuto del file di distribuzione, SDL-1.2.9.zip, troveremo un altro archivio (VisualC.zip) contenente la solution di compilazione sdl.sln. Questa solution è stata realizzata per una vecchia versione di Visual Studio: infatti, eseguendo il file ci verrà richiesta se vogliamo effettuarne la conversione ma possiamo tranquillamente confermare e proseguire in quanto è in questa operazione non darà problemi. Convertita e aperta la solution possiamo effetturare la compilazione in modalità Release, ottenendo le libreria SDL.lib e SDLmain.lib che troveremo rispettivamente nelle cartelle ..\SDL\Release e ..\SDLmain\Release. A questo punto basterà copiare questi file all'interno della cartella lib presente nella directory in cui abbiamo estratto la distribuzione; in questa cartella avremo quindi le tre librerie di runtime necessarie al proseguio, animadead.lib, SDL.lib e SDLmain.lib.

ESEMPIO: IL BOXMAN

Per verificare che tutto è andato a buon fine, apriamo la solution dell'esempio demo fornito con la distribuzione e proviamo a compilarlo. Noteremo subito che c'è qualcosa che non va: Visual Studio ci segnalerà subito l'assenza di alcuni file, tutti relativi all'SDL. Questo problema discende sempre dall'assenza nella distribuzione della libreria SDL, ma esiste una soluzione: basta andare sulle proprietà del progetto demo attraverso il *Solution Explorer* di Visual Studio, scegliere la voce *Additional Include Directories* nel campo C/C++ delle *Configuration Propeties* e premere sul pulsante sulla riga. Nella finestra aperta, aggiungete una nuova linea (il pulsante con il simbolo della cartella), indicate il percorso della cartella include presente nella distribuzione dell'SDL ed il gioco è fatto. Se riprovate a compilare la solution, questa volta Visual Studio segnelerà qualche warning ma nessun errore, segnale che questa volta tutto è andato a buon fine. Nella cartella demo troveremo adesso l'eseguibile *demo.exe*. La libreria dinamica *SDL.dll* era già stata caricata in precedente al momento di estratte l'archivio conte-

COMPILARE ANIMADEAD

Per ottenere una libreria funzionante di Animadead è necessario prima compilare i sorgenti disponibili. L'operazione è abbastanza semplice anche se strettamente legata alla piattaforma su cui si sta lavorando. Ad esempio, lavorando sotto Windows e avendo a disposizione Visual Studio .Net 2003 si può utilizzare direttamente la solution di compilazione rilasciata insieme alla distribuzione. Estraiamo l'archivio *VS.NET_2003.zip* nella stessa cartella in cui abbiamo estratto la distribuzione di Animadead, eseguiamo la solution *animadead.sln* e lanciamo una compilazione in modalità *Release* al termine della quale troveremo nella cartella della distribuzione una nuova directory *lib* contenente la libreria di runtime *animadead.lib.*

nete la solution di compilazione di *Animadead*. Dobbiamo ricordarci sempre che questa libreria va copiata sempre nella cartella in cui si troverà l'eseguibile dell'applicazione in quanto è necessaria al suo funzionamento. Lesempio demo usa un modello di una figura umanoide di cui possiamo controllarne il movimento in un campo usando i tasti *a, s, d* e *w;* tenendo premuto il pulsante sinistro del mouse si può controllare il movimento delle braccia spostando il puntatore mentre tenendo il pulsante destro si può controllare la posizione delle luci di campo. Dando un'occhiata ai modelli usati nell'esempio troviamo una serie di file *.ada* ed un unico *.adm*. Quest'ultimo contiene le meshes del modello, la struttura "fisica" della figura, composta dall'ossatura e dal rivestimento ad essa legata. I vari *.ada* invece corrispondono alle animations, cioè alle varie pose per unità di tempo controllabili dall'utente. Quello che è veramente disarmante è l'estrema semplicità con cui lavora Animadead sul modello, attraverso pochi passi.

IL MODELLO

Partendo da un'animazione già esistente, ottenuto seguendo la tecnica vista prima, la prima cosa da fare è quello di costruirci una classe modello con cui interagire. Questa classe modello non deve far altro che caricare le informazioni sulle *meshes*

```
model->AddMesh(DEMOSRCDIR
                    "models/boxman/boxman.adm");
```

e sulle *animations* previste

```
anim[0] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkNW.ada");
anim[1] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkN.ada");
anim[2] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkNE.ada");
anim[3] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkW.ada");
anim[4] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/Walk0.ada");
anim[5] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkE.ada");
anim[6] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkSW.ada");
anim[7] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkS.ada");
anim[8] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/WalkSE.ada");
anim[9] = model->AddAnim(DEMOSRCDIR
                    "models/boxman/twistflail.ada");
```

e gestire il *Draw* e l'*Update* in base ai comandi impartiti dall'utente. Ad un comando viene associata una "reazione" del modello, cioè una posa e quindi un'animazione: il programmatore deve soltanto richiamare la funzione *Calculate* sull'animazione passando la posa

```
anim[i]->Calculate(model->renderPose);
```

L'ENGINE

All'Engine viene invece delegata la gestione della posizione del modello e della scena in base ai comandi dell'utente. La cosa avviene regolando dei parametri di coordiante spaziali *x, y, z* e la scelta della posa sul modello. Ad esempio, la pressione del tasto *s* fa indietreggiare la figura. Come avviene ciò? L'engine cattura i comandi dell'utente tasto e imposta il valore di due variabili *user_x* e *user_y* sul modello *boxman* e poi chiede al modello di aggiornarsi *boxman->Update*. Il modello sa quale delle n animazioni previste deve prendere in funzione del valore *user_x* e *user_y* e richiama su di essa il metodo *Calculate* usando la *renderPose* del modello per dare la nuova posa. Fatto ciò il modello aggiorna la sua posizione agendo semplicemente sulle coordinate *x, y* e *z* per ottenere l'effetto movimento.

ESEMPIO: L'EAGLE

Passare da un modello ad un altro è veramente semplice, basta modificare veramente poche cose. Prendiamo l'esempio realizzato usando Animadead allegato alla rivista *(demo_eagle.zip)*: in questo caso il modello è quello di un'aquila realizzata mediante la tecnica dell'armatura. Da una parte abbiamo sempre le *meshes* (il file *eagle.adm*) e dall'altra le animations associate (*eagle.ada*). Realizziamo la classe modello nel modo seguente:

```
#include "eagle.h"
#include "animadead.h"
#include "SDLtexture.h"
//#include "vector3.h"
#ifdef WIN32
#include <windows.h>
#endif // WIN32
#ifdef HAVE_APPLE_OPENGL_FRAMEWORK
#include <OpenGL/gl.h>
#else
#include <GL/gl.h>
#endif
#include <math.h>
// this is used for linux builds so that the data can
// still be accessed if the
// project is built in another directory.
#ifndef DEMOSRCDIR
```

OPENGL

Da Open Graphics Library, ossia libreria aperta di grafica) è una specifica che definisce una API per più linguaggi e per più piattaforme per scrivere applicazioni che producono computer grafica 2D e 3D. Attraverso l'interfaccia si possono disegnare complesse scene tridimensionali a partira da semplici primitive. È usato per sviluppare nell'industria dei videogiochi (nella quale compete con Direct3D su Microsoft Windows), per applicazioni di CAD, realtà virtuale, e CAE. È lo standard di fatto per la computer grafica 3D in ambiente Unix.

```
#define DEMOSRCDIR
#endif // DEMOSRCDIR
#include <string>
using namespace std;
Eagle::Eagle() : x(0), y(0), z(0)
{ texture_loader = new TextureLoader();
   model = new SuperModel(ad::Model::DEFORMABLE,
                                          texture_loader);
   //anim = model->AddAnim(DEMOSRCDIR "bird/still.ada");
   anim = model->AddAnim(DEMOSRCDIR
                                "model_eagle/eagle.ada");
   model->AllocateRenderPose();
   //model->AddMesh(DEMOSRCDIR "bird/bird.adm");
   model->AddMesh(DEMOSRCDIR
                              "model_eagle/eagle.adm"); }
Eagle::~Eagle()
{  if (model) delete model;
   if (texture_loader) delete texture_loader; }
void Eagle::Draw()
{  glPushMatrix();
    glTranslatef(x,y,z);
    glRotatef(heading, 0,1,0);
            model->Draw();
            glPopMatrix(); }
void Eagle::Update(float time)
{  model->Update(time);
    anim->Calculate(model->renderPose);
}
```

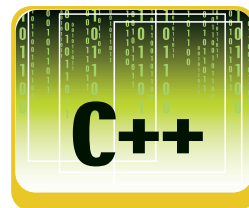
L'update del model non farà altro che passare alla posa successiva. Nell'engine gli unici *Event* a cui facciamo reagire il modello è sul movimento del mouse (se il pulsante sinistro è tenuto premuto l'aquila ruoterà nella direzione del movimento, se invece ad essere premuto si sposterà la posizione della luce di campo) e all'*ESC* (che farà terminare l'applicazione)

```
void Engine::Events()
{  static bool mbuttondown1 = false;
    static bool mbuttondown2 = false;
    static bool mbuttondown3 = false;
    static bool altmod = false;
    SDL_Event event;
    while( SDL_PollEvent(&event) )
    {  switch( event.type )
       {  case SDL_KEYDOWN:
           if (event.key.keysym.sym == SDLK_RALT ||
                   event.key.keysym.sym == SDLK_LALT)
            { altmod = true; }
            else
            HandleKey(event.key.keysym.sym, true);
            break;
            case SDL_KEYUP:
            if (event.key.keysym.sym == SDLK_RALT
                || event.key.keysym.sym == SDLK_LALT)
            {  altmod = false; }
            else
              HandleKey(event.key.keysym.sym, false);
                   break;
               case SDL_MOUSEBUTTONDOWN:
                   if (event.button.button ==
                                          SDL_BUTTON_LEFT)
                 mbuttondown1 = true;
               else if (event.button.button ==
                                         SDL_BUTTON_RIGHT)
                 mbuttondown2 = true;
               else if (event.button.button ==
                                        SDL_BUTTON_MIDDLE)
                   mbuttondown3 = true;
                   SDL_WM_GrabInput(SDL_GRAB_ON);
               break;
                 case SDL_MOUSEBUTTONUP:
               if (event.button.button ==
                                          SDL_BUTTON_LEFT)
                 mbuttondown1 = false;
               else if (event.button.button ==
                                         SDL_BUTTON_RIGHT)
                 mbuttondown2 = false;
               else if (event.button.button ==
                                        SDL_BUTTON_MIDDLE)
               mbuttondown3 = false;
               break;
                 case SDL_MOUSEMOTION:
                 if (altmod)
               { if (mbuttondown1) Rotate(
                   event.motion.xrel, event.motion.yrel);
                 if (mbuttondown2) Zoom(
                   event.motion.xrel, event.motion.yrel);
                 if (mbuttondown3) Pan(event.motion.xrel,
                                        event.motion.yrel);
                 } else
                { if (mbuttondown1) Rotate2(
                   event.motion.xrel, event.motion.yrel);
                   if (mbuttondown2) RotateLight(
                  event.motion.xrel, event.motion.yrel);}
                 break;
                   case SDL_QUIT:
                   throw(0);
                 break; } }
}
```

Compilando la solution e mandando in esecuzione il file *demo_eagle.exe*, presente nella cartella principale, l'aquila prenderà il volo.

CONCLUSIONI

Abbiamo visto con che semplicità possiamo integrare l'engine in una qualsiasi applicazione scrivendo pochissime linee di codice e senza troppi sforzi di programmazione. L'importante è avere un buon modello su cui lavorare e ricordarsi di aggiungere la libreria dinamica dell'SDL (e relativo file di licenza) ad ogni nostra distribuzione.

Antonino Panella

PLUG-IN

Animadead supporta un formato proprietario in cui le informazioni dell'animazioni del modello sono separate in meshes (la struttura geometrica) e le animations (i movimenti). Al momento, con la versione 2.0 è disponibile soltanto un plug-in che consente di utilizzare Maya come ambiente di sviluppo dei modelli, ma altri sono in fase di realizzazione, inclusi i plug-in per 3D Studio Max e Lightwave e saranno presto disponibili con la versione 3.0 di Animadead.

SUL WEB

Ultima release di Animadead:
http://animadead.sourceforge.net

La libreria SDL:
http://www.libsdl.org

L'ambiente Dev-C++:
http://www.bloodshed.net/dev/index.html

AGGREGATORE DI FEED RSS PER CELLULARI

VEDREMO COME PROGETTARE E SVILUPPARE UN AGGREGATORE DI CONTENUTI PUBBLICATI TRAMITE FEED RSS CHE POTRÀ ESSERE INSTALLATO SUI COMUNI TELEFONI CELLULARI MODERNI

So cosa una buona parte di voi starà pensando: *"Di nuovo un articolo sui feed RSS?!"*. Infatti, la maggior parte di noi programmatori è stufa di sentire parlare di questo "dialetto" dell'XML. Questo articolo, però, si contraddistingue dagli altri per un importante motivo. Vedremo, infatti, come sviluppare una semplice ed efficace applicazione orientata ai dispositivi mobili che ci permetterà di ricevere e visualizzare i contenuti dei feed RSS su un comunissimo cellulare a patto che supporti Java.

LA NOSTRA APPLICAZIONE

Emerge dalla premessa che la nostra applicazione sarà di tipo client/server. Il client sarà rappresentato da un'applicazione sviluppata utilizzando J2ME e che sarà installata su un cellulare. Il server sarà costituito da un paio di file scritti in PHP che saranno installati su un Web server e che si occuperanno di recuperare le notizie dai feed RSS. La **Figura 1** dovrebbe chiarire il concetto.

Fig. 1: **Comunicazione tra client e server. Il client richiede un tipo di notizia al server e quest'ultimo estrae, dal feed RSS adatto, le news richieste**

LATO SERVER

Come accennato, il lato server sarà sviluppato in PHP. Sul numero 95 di ioProgrammo (quello di Ottobre 2005 per intenderci) è stato pubblicato un articolo in cui venivano evidenziati i vantaggi che era possibile trarre dall'interazione J2ME/PHP al posto di J2ME/J2EE. Per tale ragione, non entreremo di nuovo nel dettaglio dell'argomentazione. Per chi avesse perso quel numero della rivista (e questo è male!) ripetiamo solo i vantaggi principali:

- PHP è ampiamente supportato dalla stragrande maggioranza dei Web server.
- PHP è open source e quindi dispone di una vasta community di programmatori che pubblicano codice con filosofia open.
- La curva di apprendimento di PHP è, senza alcuna ombra di dubbio, più bassa rispetto a J2EE.

Detto ciò, se per qualsiasi motivo voleste utilizzare un'altra tecnologia lato server, al posto di PHP, siete liberi di farlo poiché per la nostra applicazione non farà alcuna differenza. Questo perché client e server comunicheranno attraverso un protocollo conosciuto da entrambi ed ampiamente supportato: HTTP. Se vi trovate più a vostro agio con l'ambiente .NET, potrete anche scrivere il lato server in ASP.NET! La cosa più importante è che il server restituisca al client una stringa in un particolare formato che vedremo tra poco.
Lo script PHP atto a ricevere la richiesta dal client è *RSS.php*; ecco il codice:

```
<?php

require("./RSSParser.php");
$channels = array("Calcio" => "http://rss.kataweb.it
        /news/calcio/index.rss","Sport" =>"http://rss
                .kataweb.it/news/sport /index.rss",
        "Informatica" => "http://punto-informatico.it
                /fader/pixml.xml", "Ultima Ora" =>
                "http://rss.kataweb.it/news/nazionali
                                        /index.rss");
$channel = $_GET["channel"];
$url = $channels["$channel"];
```

```
$news = parseRSS($url);
echo $news;
?>
```

L'array *$channels* effettua il mapping tra il tipo di notizia (Calcio, Sport, ecc.) ed il feed RSS relativo. Il file *RSSParser.php* contiene il codice atto ad effettuare il parsing del feed RSS. Esso, per ragioni di spazio, non sarà esaminato in dettaglio in quanto è un argomento già trattato e ritrattato. Quello che è necessario sapere, in questo contesto, è che dato un tipo di notizia, i nostri script lato server estrarranno titoli e descrizioni di ogni news presente nel feed RSS corrispondente e li restituiranno al client nel seguente formato:

```
    <t>Titolo del primo articolo</t>
    <d>Descrizione del primo articolo</d>
    <t>Titolo del secondo articolo</t>
    <d>Descrizione del secondo articolo</d>
    ...
```

In particolare, è la funzione parseRSS (definita in *RSSParser.php*) ad effettuare il parsing e restituire la stringa appena vista. Abbiamo deciso di "restringere" le news in questo modo per due motivi principali:

- Minore è il numero di byte che il cellulare riceve dal server e minore è il costo dell'operazione (questo salvo contratti particolari col gestore di rete mobile).
- L'applicazione è più performante se la quantità di byte da scaricare è inferiore (questo a prescindere da qualsiasi contratto particolare col gestore di rete mobile!).

Chiaramente sarà compito del client, ovvero dell'applicazione J2ME, estrarre le informazioni di interesse da quella stringa e mostrarle in maniera convenevole all'utente.

LATO CLIENT

L'applicazione lato client dovrà sostanzialmente essere in grado di:

- Inviare una richiesta al server comunicando il tipo di news desiderate.
- Ricevere le news in risposta dal server e visualizzarle all'utente.

Ovviamente, ci sarà un'interfaccia grafica che permetterà appunto all'utente di selezionare il tipo di news e visualizzarle sul cellulare. La **Tabella 1** illustra le classi utilizzate dall'applicazione e le relative responsabilità. La classe *News* è abbastanza semplice ed inizia come segue:

Classe	Responsabilità
News	Rappresentare una news
NewsFetcher	Comunicare col server e recuperare le news
NewsParser	Estrarre le news dalla stringa ricevuta dal server
WirelessNews	GUI dell'applicazione, ovvero la midlet

Tabella 1: Classi e relative responsabilità.

```
class News
{
    String title;
    String description;

    public News(String title, String description)
    {
        this.title = title;
        this.description = description;
    }
    [...]
}
```

Come si può vedere una news è rappresentata da titolo e descrizione. Il resto della classe è costituito dai classici metodi *"getter"* e *"setter"* relativi a titolo e descrizione.
Nei paragrafi successivi saranno esaminate le restanti classi che costituiscono la nostra applicazione. Chiaramente, non sarà riportato tutto il codice; lo troverete, in ogni modo, nel CD allegato alla rivista.

LA CLASSE WIRELESSNEWS

Come già accennato, la classe *WirelessNews* si occupa di gestire l'interfaccia grafica *(GUI)*. La nostra applicazione si può trovare in una dei seguenti stati:

A) Lista contenente i differenti tipi di news (**Figura 2**).

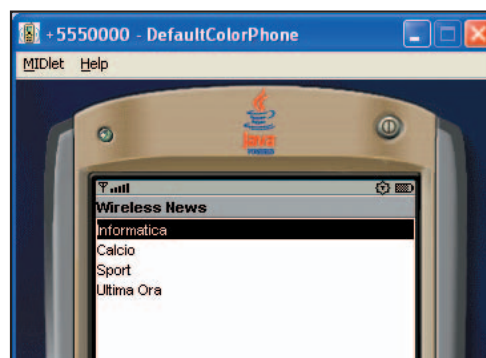
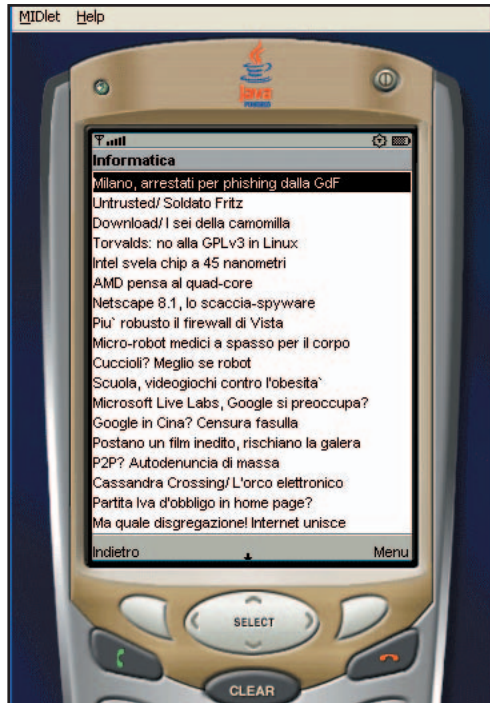



Fig. 2: La nostra applicazione

B) Schermata che illustra una lista contenente i titoli delle news scaricate dal Web (**Figura 3**).

C) Form in cui viene visualizzata la notizia completa, ovvero titolo e descrizione (**Figura 4**).

In sostanza, all'avvio dell'applicazione viene visualizzata la lista contenente i vari tipi di news: *Informatica, Calcio, Sport,* ecc. come mostrato in **Figura 2**. Selezionando un tipo di news si avvia il processo di "fetching" delle news vere e proprie. Ovvero si comunica allo script *RSS.php,* che si trova sul Web server, il tipo di notizia desiderata. Lo script effettua il parsing del feed RSS corrispondente e restituisce al client una stringa nel formato già visto.
A questo punto, l'applicazione sul cellulare effettua il parsing della stringa ricevuta dal server e visualizza, come in **Figura 3**, una lista contenente i titoli delle notizie ricevute. Da questa schermata è possibile selezionare uno dei titoli visualizzati in modo tale da poter leggere la notizia completa, come illustrato in **Figura 4**.
Ecco le prime righe di codice della classe *WirelessNews*:

```
public class WirelessNews extends MIDlet
                         implements CommandListener
{

   private Display display;
   private List channelsList;
   private String[] channels =
   {
     "Calcio", "Sport", "Informatica",
                                    "Ultima Ora"
   };

   private List titlesList;
   private Form fmDescription;
   private News[] news;
   private Command cmSelect;
   private Command cmExit;
   private Command cmBack;
   private Command cmMain;
   [...]
```

Come detto, *WirelessNews* è la midlet della nostra applicazione e come tale estende la classe *MIDlet.*
Inoltre, *WirelessNews* implementa l'interfaccia *CommandListener* la quale ci permette di gestire le azioni scelte dall'utente. L'oggetto display ci serve per gestire il display del dispositivo. La lista *channelsList* è utilizzata per visualizzare i vari tipi di news rappresentati dall'array di stringhe *channels*. Poi abbiamo la lista *titlesList* che verrà riempita con i titoli delle news scaricate. Il form *fmDescription* è usato per visualizzare all'utente la notizia completa, composta da titolo e descrizione. Le news scaricate dal server vengono inserite in un array di oggetti di tipo News. Le quattro dichiarazioni che seguono, sono i comandi che saranno utilizzati per gestire le azioni scelte dall'utente. Vediamo, ora, come vengono gestiti questi comandi all'interno del metodo *commandAction*, l'unico definito dall'interfaccia *CommandListener.*

```
public void commandAction(Command c,
                                   Displayable d)
{

   //comando Esci
   if(c == cmExit)
   {
      destroyApp(false);
```

```
        notifyDestroyed();
    }

    //il display è la lista contenente i tipi di news
    if(d == channelsList)
    {
      //recupera l'indice del tipo di news selezionato
      int channelIndex =
                        channelsList.getSelectedIndex();

      //istanzia la classe che recupera le news
      NewsFetcher nf = new NewsFetcher(
                        channels[channelIndex], this);

      //mostra un alert indicante l'inizio del
                                              download
      showAlert("Download in corso...attendere
                                          prego", d);

      //avvia il download vero e proprio
      nf.start();
    }

    //il display è la lista contenente
                        i titoli delle news scaricate
    if(d == titlesList)
    {
      // comando Indietro
      if(c == cmBack)
        display.setCurrent(channelsList);
        // comando Scegli ->
                        mostra la notizia completa
        if(c == cmSelect)
          showDescription(news[
                        titlesList.getSelectedIndex()]);

    }

    //il display è il form atto a
                        visualizzare la notizia completa
    if(d == fmDescription)
    {
        //comando Indietro
        if(c == cmBack)
          display.setCurrent(titlesList);
          //comando Canali (ovvero i tipi di news)
          if(c == cmMain)
          {
            titlesList = null;
            display.setCurrent(channelsList);
          }

    }
}
```

Le azioni da compiere variano secondo il display in cui si trova l'applicazione ed il comando selezionato. L'unica eccezione la fa il comando *cmExit* il quale termina l'applicazione a prescindere dal display. Se il display è la lista dei tipi di news, ovvero *channelsList*, allora se non è stato scelto *cmExit* l'altra unica possibilità è che sia stato selezionato *cmSelect*, poiché sono gli unici comandi associati a questo display. In quest'ultimo caso viene istanziato un oggetto di tipo *NewsFetcher* (che vedremo tra poco) passando al costruttore il tipo di news selezionata, ad esempio *Informatica*, e un riferimento alla midlet chiamante *(this)*. In seguito, viene visualizzato un alert che indica l'inizio del download e viene avviato il fetching delle news in un thread separato. Se il display, invece, è la lista contenente i titoli delle news, allora la selezione del comando *cmSelect* causa la visualizzazione del form contenente la notizia completa, ovvero titolo e descrizione. Se, invece, il comando scelto è *cmBack*, l'applicazione visualizzerà il menu principale, ovvero la lista contenente i tipi di news. Dal form contenente la news completa è possibile tornare al menu principale o al display precedente, ovvero quello contenente i titoli delle notizie scaricate.

LA CLASSE NEWSFETCHER

La responsabilità della classe *NewsFetcher* è di collegarsi al server, comunicare il tipo di notizia desiderato e scaricare le news vere e proprie. Lo script PHP lato server si occuperà, poi, di fare il parsing del feed RSS adatto e restituire le notizie come una stringa della forma:

```
<t>Giocatore Alfa in prestito alla squadra
                                        Beta</t>
<d>L'attaccante Alfa va alla Beta per una cifra
                        pari a 1.000.000 di euro</d>
<t>Infortunio per il portiere della Gamma</t>
<d>Delta, portiere della Gamma, starà fermo 3
        settimane in seguito ad un'uscita azzardata
                          sull'attaccante Omega</d>
```

Sarà compito della classe *NewsParser*, che vedremo tra poco, estrarre da questa stringa i titoli (contrassegnati da *<t>...</t>*) e le descrizioni (contrassegnate da *<d>...</d>*) delle news. La midlet *WirelessNews* provvederà, poi, a visualizzarle in formato *"user-friendly"* all'utente.
Ecco l'inizio della classe *NewsFetcher:*

```
class NewsFetcher implements Runnable
{
  private String channelName;
```

```
private String URL = "http://localhost:8000
                              /WirelessNews/RSS.php";
private WirelessNews midlet;
private String news;
```

Come si può vedere, *NewsFetcher* implementa l'interfaccia *Runnable* in modo da effettuare il download delle news in un thread separato, evitando così di "congelare" l'applicazione durante il recupero delle news. Gli attributi della classe sono costituiti da:

- una stringa, *channelName*, rappresentante il tipo di news desiderate;
- l'URL che punta allo script PHP responsabile di recuperare le news;
- un reference alla midlet *WirelessNews;*
- una stringa atta a contenere l'insieme di news scaricate.

L'unica cosa da notare è il formato dell'URL. Nel mio caso ho impostato il Web server Apache in ascolto sulla porta 8000 dato che la porta di default, che è 80 per HTTP, era già "occupata" da IIS. Chiaramente, una volta caricato lo script PHP online, dovrete cambiare l'URL di conseguenza per farlo puntare all'indirizzo corretto. Ad esempio, l'URL potrebbe essere:

http://www.nomesito.it/RSS.php

Il metodo principale della classe *NewsFetcher* è *getNews* il cui codice è:

```
private void getNews() throws IOException
{
  InputStream is = null;
  StringBuffer sb = new StringBuffer();
  HttpConnection http = null;
  try
  {
       //aggiunge il tipo di news desiderato
                                        all'URL
       URL += "?channel=" + channelName;
       //sostituisce i caratteri non permessi in
                                        un URL
     URL = EncodeURL(URL);
     //stabilisce la connessione
     http = (HttpConnection)
     Connector.open(URL);
     //imposta il metodo di richiesta a GET
     http.setRequestMethod(
                         HttpConnection.GET);
     //riceve il response dal server
     if(http.getResponseCode() ==
                      HttpConnection.HTTP_OK)
     {
        int ch;
            is = http.openInputStream();
            while((ch = is.read()) != -1)
            sb.append((char) ch);
        }
  }
  catch (Exception e)
  {
    System.err.println("Error: " +
                                  e.toString());
    networkError();
  }
  finally
  {
      if(is != null)
        is.close();
      if(sb != null)
        news = new String(sb);
      else
        news = new String();
      if(http != null)
        http.close();
  }

  //estrae titoli e descrizioni dalle news
  if(news.startsWith("<t>"))
  {
     //niente errori
     //solo a scopo di debug
     System.out.println(news);

     News[] n = (new NewsParser(
                              news)).parse();
     midlet.showTitles(n, true, channelName);
   }
   else
   {
     //errore. Il metodo networkError() notifica
                                        l'errore
     networkError();
   }
}
```

Questo metodo accoda all'URL la query string indicante il tipo di news desiderato. Sostituisce i caratteri non permessi in un URL (come lo spazio) con i caratteri adatti. Dopodiché, cerca di stabilire una connessione di tipo *HttpConnection* col server e costruisce uno *StringBuilder* con le news ricevute dal server.
Se il download è andato a buon fine, viene effettuato il parsing della stringa rappresentante le news tramite il metodo parse della classe *News Parser.*
Tale metodo restituisce un array di oggetti di tipo *News,* ovvero le notizie vere e proprie. Infine viene chiamato il metodo *showTitles* della classe *WirelessNews* il cui scopo è visualizzare le notizie sullo schermo del cellulare o un

Per saperne di più sui feed RSS:

http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

alert di errore in caso di fallimento.

LA CLASSE NEWSPARSER

Come già accennato, tale classe ha la responsabilità di “rimettere le cose nel giusto ordine”, ovvero di estrarre titoli e descrizioni da quella stringa “grezza” ottenuta dal server. La classe *NewsParser* è stata così codificata:

```
class NewsParser
{

   private String rawNews;
   private int index;

   public NewsParser(String rawNews)
   {
      this.rawNews = rawNews;
      this.index = 0;
    }
    public News[] parse()
    {
      Vector v = new Vector();

      //estrae le news
      while(index < rawNews.length())
      {
        //estrae il titolo
        String title = extractText(index, "t");
        //estrae la descrizione
        String description = extractText(
                                        index, "d");
         News news = new News(
                               title, description);
        v.addElement(news);
      }
        News[] ret = new News[v.size()];
        v.copyInto(ret);
        return ret;
     }

     //estrae il testo (titolo o descrizione, a
                         seconda del parametro type)
     private String extractText(int beginIndex,
                                        String type)
     {
        String startTag = "<" + type + ">";
        String endTag = "</" + type + ">";

        //trova l'indice di startTag in
                    rawNews, a partire da beginIndex
        int begin = rawNews.indexOf(
                               startTag, beginIndex);
        //avanza di 3 caratteri in modo da
                        puntare al testo vero e proprio
```

```
        begin += 3;
        //trova l'indice di endTag in rawNews, a
                                  partire da begin
        int end = rawNews.indexOf(
                                  endTag, begin);
        //aggiorna il "cursore"
        index = end + 4;
        //restituisce il testo (titolo o descrizione)
        return
        rawNews.substring(begin, end);
    }
}
```

Questa classe non richiede particolari spiegazioni poiché si tratta semplicemente di manipolazione di stringhe. Vengono estratti titoli e descrizioni delle news e viene costruito un array di oggetti di tipo *News*.

CONCLUSIONI

In quest’articolo abbiamo visto come sviluppare sia il lato client sia il lato server di un aggregatore di feed RSS per dispositivi mobili. Ovviamente, l’applicazione può essere migliorata in molti modi. Ad esempio, è possibile utilizzare RMS *(Record Management System)* per memorizzare sul cellulare le news scaricate. Oppure memorizzare, sempre sul dispositivo mobile, il mapping tra tipo di news e URL relativo al feed RSS. Questo ci permetterebbe di estendere la lista dei tipi di news “dinamicamente” senza dover ricompilare l’applicazione.
Lo scopo del presente articolo, era solo quello di dare un input al lettore il quale potrà estendere o modificare il sistema visto a proprio piacimento.
Buon lavoro dunque.

Alessandro Lacava

FEED RSS

Un feed RSS è semplicemente un file XML utilizzato, principalmente, per rappresentare un insieme di articoli.
Uno stralcio di un *feed RSS* è il seguente:

```
[...]
<item>
  <title>Titolo del primo articolo
                                   </title>
  <description>Descrizione del
            primo articolo</description>
  <link>Link al primo articolo
                           completo</link>
  [...]
</item>
<item>
  <title>Titolo del secondo articolo
                                   </title>
  <description>Descrizione del
        secondo articolo</description>
  <link>Link al secondo articolo
                           completo</link>
  [...]
</item>
[...]
```

Ogni item rappresenta un articolo e contiene, tra le altre cose, il titolo, la descrizione e il link alla news completa.

VISUAL BASIC.NET E LA GRAFICA

INTEGRARE GRAFICA E DATI DINAMICAMENTE NELLE PAGINE WEB UTILIZZANDO LE FUNZIONI GRAFICHE DEL .NET FRAMEWORK. VEDREMO COME REALIZZARE BOTTONI IN FASE DI RUN TIME

Agli sviluppatori Web capita frequentemente di trovarsi a gestire alcune parti della grafica del sito, come la realizzazione, ad esempio, dei bottoni di un menu. Se pensiamo al menu di un sito web, anche semplice, possiamo immaginare una serie di bottoni che conducono alle varie pagine come quelli raffigurati in **Figura 1**.

Fig. 1: ***Un tipico menu di un sito web***

Ciò comporta necessariamente di dover elaborare i bottoni con un programma di grafica (come ad esempio Adobe Photoshop) assegnando ad ogni immagine un testo.
L'operazione, pur non complessa, è spesso impossibile da realizzare per siti con un menu molto nutrito e magari con bottoni il cui testo cambia molto spesso (perché magari proveniente da un database aggiornabile dall'utente).
La conseguenza spesso è che per siti particolarmente articolati si rinuncia all'uso di grafica per i menu per far ricorso soltanto alla formattazione offerta dai fogli di stile.
Intendiamoci, anch'io sono per un uso più limitato possibile delle immagini nella pagina web, solo che preferisco che questa sia una scelta stilistica, non un vincolo. E poi, diciamolo francamente, a volte un tocco di colore, soprattutto nei siti “vetrina”, non guasta.
Il problema è quindi come fare ad adattare un modello di un elemento grafico, ad esempio un bottone, ad un testo generato dinamicamente. Per essere più chiari: abbiamo un template grafico come quello in **Figura 2** e vogliamo che il server “ci scriva sopra” il testo per noi.

Fig. 2: ***Il template vuoto del nostro bottone***

La soluzione, in ambiente ASP.NET, sta nell'uso delle librerie grafiche offerte dal Framework. Vedremo quindi com'è possibile, partendo da un template grafico di base, far sì che l'immagine finale, composta da un bottone e da un testo, venga generata dinamicamente.

IL NOSTRO PROGETTO

In Visual Studio 2003 o, se preferite in SharpDevelop, impostiamo un nuovo progetto di tipo libreria come possiamo vedere in **Figura 3**.

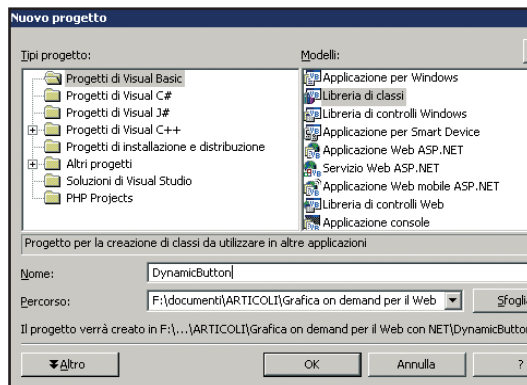



Fig. 3: ***Il nuovo progetto di libreria***

A questo punto dobbiamo aggiungere i riferimenti necessari alle librerie: System.Web e System.Drawing; la prima ci fornirà le classi per interagire con l'ambiente web mentre la seconda ci consentirà di usare la grafica GDI+.

```
Imports System.Web
Imports System.Drawing
Imports System.Drawing.Drawing2D
```

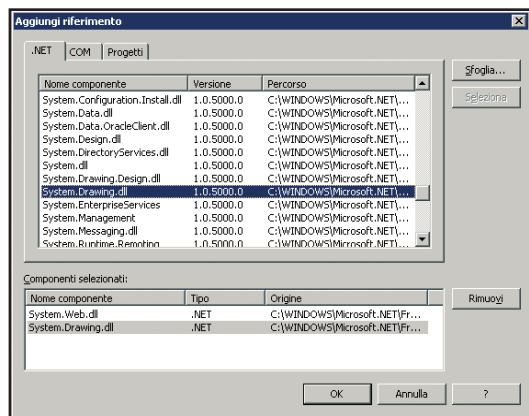



Fig. 4: ***Aggiunta dei riferimenti System.Web e System.Drawing al progetto***

Una volta preparato così l'ambiente di lavoro, passiamo a scrivere un po' di codice : aggiungiamo al progetto una classe che chiameremo GraphicButton:

```
Public Class GraphicButton
    Implements IHttpHandler

    Public ReadOnly Property IsReusable() As Boolean
          Implements System.Web.IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

    Public Sub ProcessRequest(ByVal context As
                System.Web.HttpContext) Implements
            System.Web.IHttpHandler.ProcessRequest

    End Sub
End Class
```

Questa classe rappresenta l'impalcatura della nostra applicazione. Notiamo subito che implementa un'interfaccia, *IHttpHandler*, che gestisce il processo *Request/Response* di un flusso http.
In particolare la proprietà IsReusable, impostata su False, sta ad indicare che il server deve processare nuovamente ogni richiesta, mentre il metodo ProcessRequest, fornisce, con il parametro context, l'oggetto necessario per accedere a Request e Response.
In pratica possiamo vedere il metodo ProcessRequest come il "Main" della nostra applicazione, il suo punto d'ingresso.
Per completare l'infrastruttura della classe dichiariamo due variabili private a livello di classe che conterranno gli oggetti Request e Response che rappresentano appunto la richiesta del client e la risposta del server (gli stessi oggetti noti peraltro in ASP e ASP.NET):

```
Private Request As HttpRequest
Private Response As HttpResponse
```

Questi oggetti vengono valorizzati, all'interno del metodo ProcessRequest utilizzando l'oggetto context:

```
Me.Request = context.Request
Me.Response = context.Response
```

UN TOCCO DI GRAFICA

A questo punto occorrerà sviluppare il vero e proprio codice della classe ma prima , con un editor di immagini, disegniamo il nostro bottone vuoto.

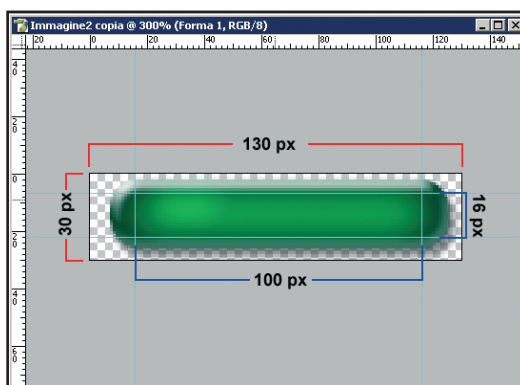



Fig. 5: ***Il bottone disegnato in Photoshop, in evidenza le dimensioni totali e il rettangolo che contiene il testo***

Come possiamo notare in figura 5 abbiamo annotato le misure del bottone ed in particolare le dimensioni del rettangolo interno (100 x16 pixel) che deve contenere il nostro testo.
Questa immagine sarà salvata in un formato non compresso (ad es. BMP) nella cartella del progetto con un nome convenzionale: btnTemplate.bmp.
Torniamo adesso al nostro codice. Inseriamo nel corpo della una proprietà che ci permette di caricare la nostra Bitmap:

```
    Private _Template As Bitmap
    Private ReadOnly Property Template() As Bitmap
        Get
            If _Template Is Nothing Then
                _Template = New Bitmap(
                    Request.MapPath("btnTemplate.bmp"
                                        ) , True)
            End If
```

IL PROGRAMMA DI GRAFICA

Come programma di grafica web abbiamo utilizzato l'insuperato Photoshop. Chi non dispone di questo strumento può utilizzare qualsiasi altro programma di questo tipo.
Noi consigliamo l'open source GIMP che potete trovare in:
http://www.gimp.org

```
            Return _Template
        End Get
    End Property
```

Notiamo qui come il caricamento del file avvenga risalendo alla path completa del file attraverso il metodo MapPath dell'oggetto Request (non dimentichiamoci che siamo in ambiente Web e che la path sarà relativa alla radice del sito).
A questo punto scriviamo il metodo, che chiameremo DisegnaBottone, che svolge tutto il lavoro di scrivere il testo sul bottone.

```
    Private Sub DisegnaBottone()
        Dim rettangoloBottone As RectangleF =
                    Template.GetBounds(GraphicsUnit.Pixel)
        Dim rettangoloTesto As RectangleF = New
                    RectangleF(rettangoloBottone.X + 16,
                        rettangoloBottone.Y + 7, 100, 16)
        Dim testo As String = Request.QueryString("s")
        If testo = "" Then testo = "prova"
        Dim coloreTesto As Color = Color.FromArgb(
                                        200, Color.White)
        Dim pennelloTesto As SolidBrush = New
                                    SolidBrush(coloreTesto)
        Dim formatoTesto As New StringFormat(
                        StringFormatFlags.FitBlackBox)
        formatoTesto.Alignment =
                                StringAlignment.Center
        formatoTesto.LineAlignment =
                                StringAlignment.Center
        Dim carattere As New Font("verdana", 14,
                                    GraphicsUnit.Pixel)
        Dim g As Graphics = Graphics.FromImage(
                                            Template)
        g.DrawString(testo, carattere, pennelloTesto,
                        rettangoloTesto, formatoTesto)
        g.Dispose()
        Me.Response.ContentType = "image/jpeg"
        Template.Save(Me.Response.OutputStream,
                        Imaging.ImageFormat.Jpeg)
    End Sub
```

Seguiamo passo passo il codice.
Per prima cosa ricaviamo i due rettangoli; dell'area occupata dal bottone (prendendola dalla bitmap di template) e del rettangolo interno che ospita il testo (con le misure che avevamo già visto in Photoshop):

```
Dim rettangoloBottone As RectangleF =
                Template.GetBounds(GraphicsUnit.Pixel)
Dim rettangoloTesto As RectangleF = New
                RectangleF(rettangoloBottone.X + 16,
                    rettangoloBottone.Y + 7, 100, 16)
```

Prendiamo poi il testo da scrivere che deve essere passato come parametro nella stringa della URL (o in mancanza essere valorizzato con la stringa "prova"):

```
Dim testo As String = Request.QueryString("s")
If testo = "" Then testo = "prova"
```

Impostiamo poi il colore del testo e da questo ricaviamo il pennello grafico (notiamo come al colore, Bianco, sia stata aggiunta una piccola trasparenza per fondersi meglio con la bitmap):

```
Dim coloreTesto As Color = Color.FromArgb(200,
                                        Color.White)
Dim pennelloTesto As SolidBrush = New
                                SolidBrush(coloreTesto)
```

Impostiamo inoltre il carattere del testo e il formato della stringa che saranno rispettivamente: verdana a 14 pixel e centrato sia verticalmente che orizzontalmente:

```
Dim formatoTesto As New StringFormat(
                    StringFormatFlags.FitBlackBox)
formatoTesto.Alignment = StringAlignment.Center
formatoTesto.LineAlignment = StringAlignment.Center
Dim carattere As New Font("verdana", 14,
                                GraphicsUnit.Pixel)
```

Otteniamo l'oggetto Graphics dal nostro template che abbiamo caricato e andiamo a scrivere la nostra stringa di testo, con i parametri impostati, nel rettangolo relativo:

```
Dim g As Graphics = Graphics.FromImage(Template)
g.DrawString(testo, carattere, pennelloTesto,
                    rettangoloTesto, formatoTesto)
g.Dispose()
```

Infine salviamo la bitmap così modificata nello Stream di output dell'oggetto Response impostando prima la proprietà ContentType di Response come "image/jpeg":

```
Me.Response.ContentType = "image/jpeg"
Template.Save(Me.Response.OutputStream,
                    Imaging.ImageFormat.Jpeg)
```

Completiamo la nostra classe semplicemente richiamando il metodo DisegnaBottone all'interno di ProcessRequest:

```
Public Sub ProcessRequest(ByVal context As
                System.Web.HttpContext) Implements
            System.Web.IHttpHandler.ProcessRequest
        Me.Request = context.Request
        Me.Response = context.Response
```

NOTA

PANORAMICA SULLE GDI

Per i programmatori VB.NET non è facile trovare molte risorse sull'uso della GDI (Graphic Device Interface) dal Framework. Per qualche strana ragione la materia sembra "appalto" dei programmatori C# quando invece le classi GDI possono essere usate in modo praticamente identico nell'uno e nell'altro linguaggio. Una buona risorsa sul web è comunque http://www.vbdotnetheaven.com/Sections/GDI+.asp

```
        DisegnaBottone()
End Sub
```

COME L’UTILIZZIAMO?

A questo punto anche i più pazienti tra di voi staranno incominciando a chiedersi: ma quando si passa all’azione?
Vi accontentiamo subito.
Ricordate che la nostra classe implementava IHttpHandler? Bene questo non è altro che il gestore “a basso livello” del meccanismo richiesta client/risposta server in un contesto HTTP (IHttpHandler è utilizzato in modo nascosto anche dalle pagine ASP.NET) quindi dobbiamo semplicemente impostare l’applicazione web e dirle che intendiamo utilizzare il tipo GraphicButton per gestire una particolare URL.
Niente paura, la cosa è più difficile a dirsi che a farsi.
Per prima cosa, nel server IIS, impostiamo una nuova directory virtuale, come vediamo in **Figura 6**.

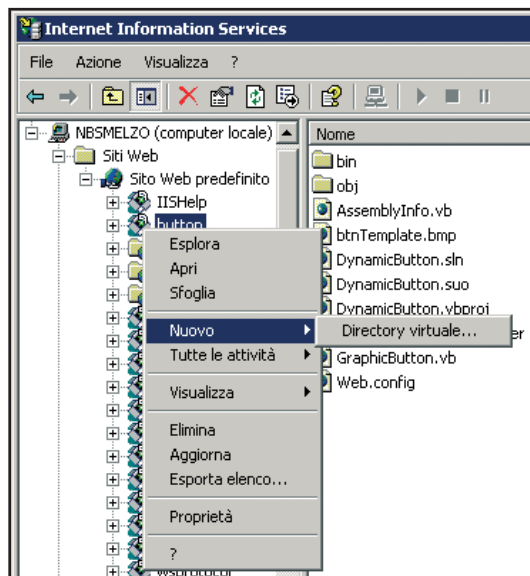



Fig. 6: ***Impostiamo una nuova directory virtuale in IIS***

Nel wizard che segue impostiamo il nome della directory in button e gli assegniamo, come path, il percorso completo della directory del nostro progetto.
A questo punto, se operiamo nel nostro computer con IIS installato, abbiamo disponibile la nostra applicazione alla URL: *http://localhost/button*.
Sì, direte voi, ma non abbiamo creato neanche una pagina ASP.NET ?
Non è necessario. Nella directory del nostro progetto (che a questo punto sarà diventata anche la root di *http://localhost/button*) creiamo, con un semplice editor di testo o con Visual Studio, un file web.config di questo tipo:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.web>
          <globalization requestEncoding="iso-8859-1"
                        responseEncoding="iso-8859-1"/>
          <httpHandlers>
          <add path="button.aspx" type=
                        "DynamicButton.GraphicButton,
                  DynamicButton" verb="*" validate=
                                      "false"></add>
          </httpHandlers>
  </system.web>
</configuration>
```

Notiamo in particolare l’elemento add all’interno di httpHandlers, in questo elemento abbiamo definito gli attributi:

- **path** – che è la url relativa all’applicazione che verrà gestita, in questo caso button .aspx che verrà richiamata come *http://localhost/ button/button.aspx.*
- **type** – che è composto da <tipo>,<assembly>, cioè il nome completo del tipo (o classe) che gestisce la richiesta (nel nostro caso GraphicButton completo del namespace DynamicButton che il progetto crea automaticamente) e del nome dell’assembly (DLL) che lo contiene (nel nostro caso anch’esso chiamato DynamicButton).
- **verb** – impostato a “*” ad indicare che il gestore è valido per tutte le richieste (POST, GET ecc...)
- **validate** – impostato a false che indica che non deve essere cercato su disco il file button.aspx ma che esso è solo una URL convenzionale.

Fig. 7: ***La nostra libreria in funzione!***

ALTERNATIVE A IIS

IIS è il web server di elezione per testare le applicazioni web create con .NET tuttavia il server non è disponibile per alcune piattaforme, come Windows XP Home. Gli utenti che non possono disporre di IIS possono installare il web server Cassini, messo a disposizione dalla stessa Microsoft su http://www.asp.net/Projects/Cassini/Download/Default.aspx?tabindex=0&tabid=1

Ed è proprio l’ultimo attributo, validate, che impostato su false ci consente di avere a disposizione l’output prodotto dalla nostra classe senza avere in realtà un file Button .aspx.
A questo punto, per provare che tutto funzioni a dovere, compiliamo il progetto e nel browser andiamo a digitare l’URL *http://localhost/button/button.aspx* e otteniamo il nostro template riempito con il testo di prova.
Facciamo quindi un’ulteriore prova aggiungendo alla URL attraverso il parametro s la stringa “home”: *http://localhost/button/button.aspx?s=home* e notiamo come il testo sul bottone sia cambiato (**Figura 8**).

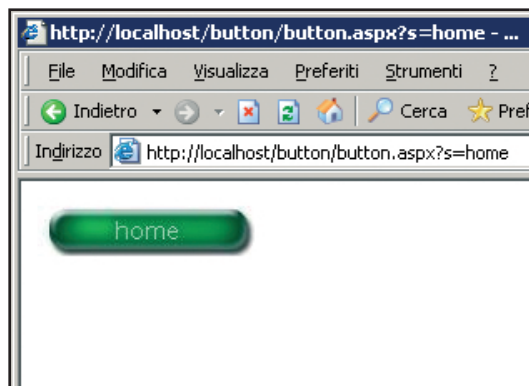



Fig. 8: ***Il testo del bottone è regolato dal parametro s della querystring***

UN PO’ DI HTML

A questo punto abbiamo capito che l’output della nostra classe viene trattato dal browser come un qualsiasi file grafico statico, per cui come tale potremmo utilizzarlo all’interno del nostro codice html.
Per fare una prova creiamo, sempre nella root del sito, un semplice file HTML chiamato menu.htm inserendo le url che puntano a button.aspx come sorgenti di immagini e cambiando soltanto il parametro s:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
                                    Transitional//EN">
<html>
<head>
<title>menu</title>
</head>

<body>
  <img src="button.aspx?s=home"><br>
  <img src="button.aspx?s=chi%20siamo"><br>
  <img src="button.aspx?s=prodotti"><br>
  <img src="button.aspx?s=l'azienda"><br>
  <img src="button.aspx?s=contatti"><br>
```

```
</body>
</html>
```

E, come risultato, possiamo ammirare la bella lista di bottoni presentata in **Figura 9**.

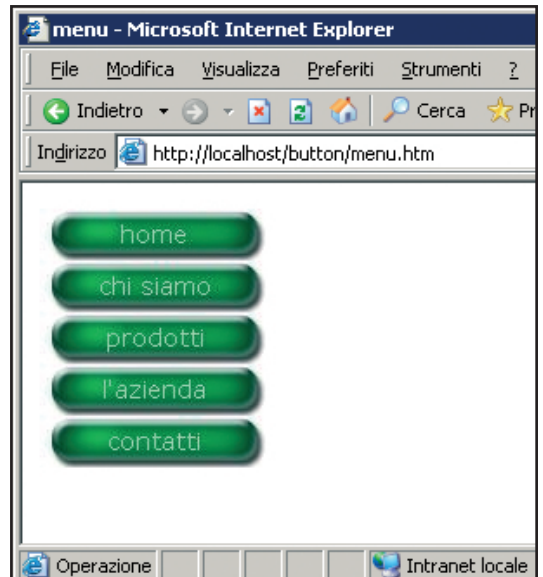



Fig. 9: ***Il nostro menu grafico costruito dinamicamente***

MARGINI DI MIGLIORAMENTO

Non vi sarà sfuggito che la nostra classe che abbiamo scritto GraphicButton in un ambiente di produzione può essere notevolmente migliorata per aumentarne flessibilità e riutilizzo.
In particolare, nel nostro esempio abbiamo parametrizzato soltanto il testo che viene valorizzato attraverso il parametro s della collection Request.Querystring, accodandolo in pratica alla url. In tal modo però si possono parametrizzare anche tutti gli altri valori che qui sono stati scritti direttamente nel codice : font, dimensioni, stile e anche la stessa immagine di base.
C’è poi, per gli amanti dell’avventura, la strada più impervia di non servirsi di un template pre-esistente, ma di disegnarlo direttamente con i metodi grafici (DrawRectangle,FillEllipse ecc...); in questo modo si possono ottenere risultati davvero flessibili ed eccellenti soprattutto quando il pulsante si deve adattare alla lunghezza del testo.
Tuttavia lo scopo dell’articolo era quello di mostrare una strada possibile per l’integrazione tra grafica e dati dinamici ed una descrizione troppo dettagliata delle funzioni GDI ci avrebbe portato fuori obiettivo.

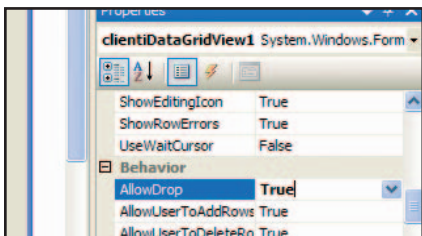
Francesco Smelzo

L’AUTORE

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop. Il suo sito web è www.smelzo.it. **Come sempre è a disposizione per ricevere suggerimenti o richieste sull’articolo all’indirizzo di posta elettronica** francesco@smelzo.it

IO PROGRAMMO BY EXAMPLE

IMPARA A PROGRAMMARE IN MODO PRATICO E DIVERTENTE, CON GLI ESEMPI PASSO PASSO CHE TI GUIDANO ALLA COSTRUZIONE DEL CODICE

Un trucco utile per copiare i dati da una tabella ad un'altra in modo semplice e immediato. Vediamo come.

Creeremo una nuova classe derivata dall'originale e sfrutteremo la proprietà DrawMode per determinare il disegno della combo

Può capitare di voler eliminare i tag presenti in un testo HTML per poterlo consultare con un editore. Vediamo come.

Si tratta di una tecnica che consente di convertire un tipo di dato in un'altro, da adottare con modalità particolari per ogni linguaggio. Vediamo come.

Utilizzeremo in modo semplice alcune DLL di sistema per raggiungere lo scopo, vedremo che è più semplice di quello che normalmente si pensa

Esistono molti comodi metodi per poter lavoare con le stringhe. Alcuni di essi hanno significati particolari. Vediamo quali.

Un numero intero è pari se è divisibile per due, in caso contrario è dispari. Sfruttiamo questa definizione per recuperare l'informazione che ci serve.

Possiamo utilizzare alcune delle caratteristiche peculiari del linguaggio. Vediamo un esempio rapido realizzato con Eclipse.

Un esempio pratico per imparare come vengono richiamate le classi all'interno di Java e come viene gestita la memoria.

VUOI INVIARE UN ESEMPIO?

Se sei un programmatore esperto ed hai risolto un problema, puoi aiutare gli altri pubblicando il tuo codice. Proponi i tuoi esempi scrivendo a ioprogrammo@edmaster.it

Come contattarci?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

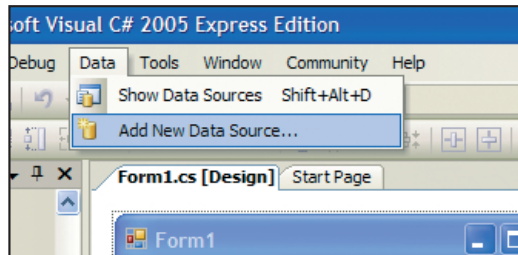
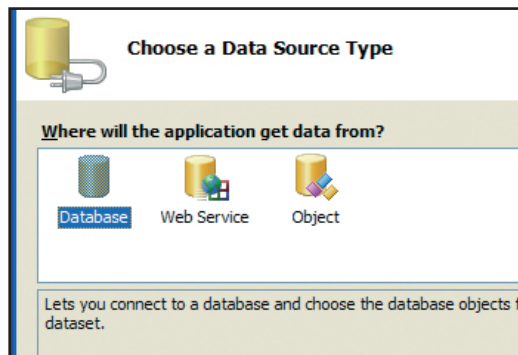
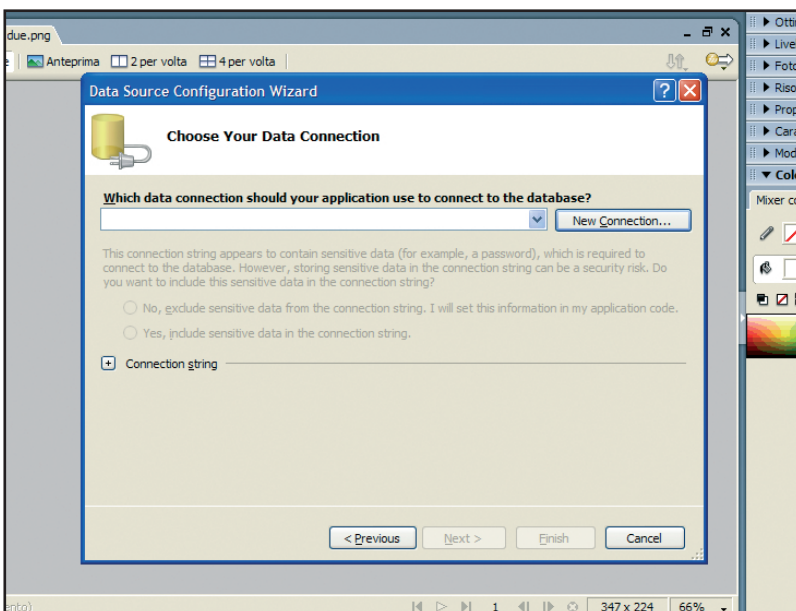
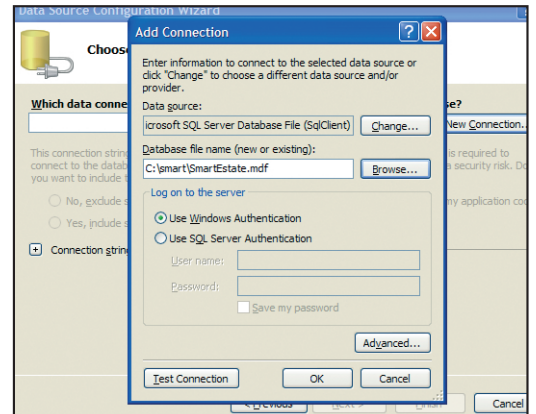
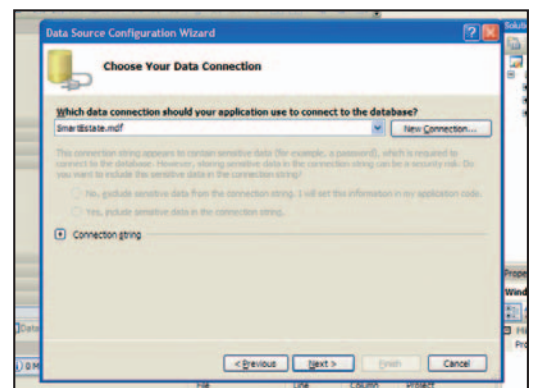
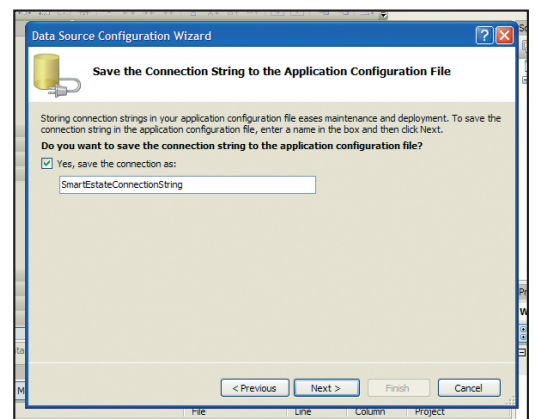
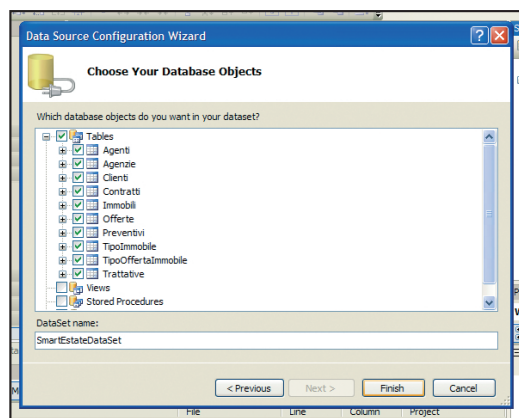
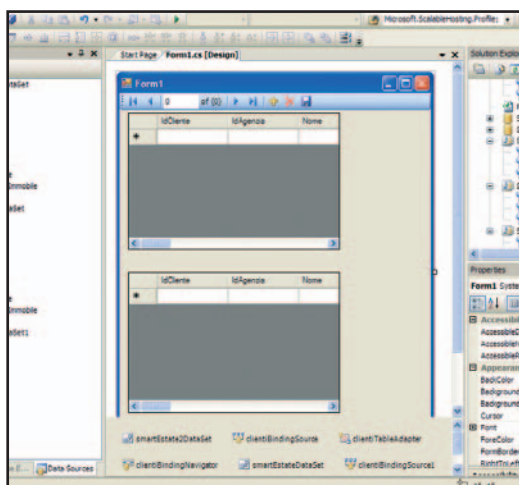
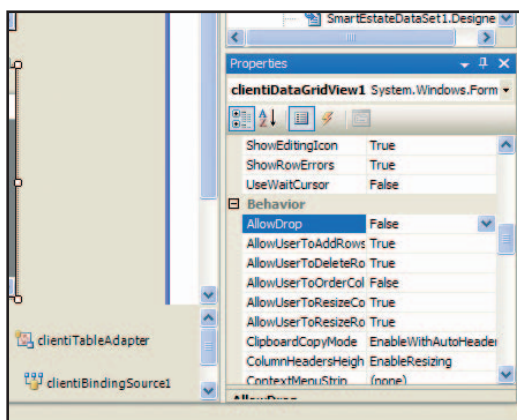
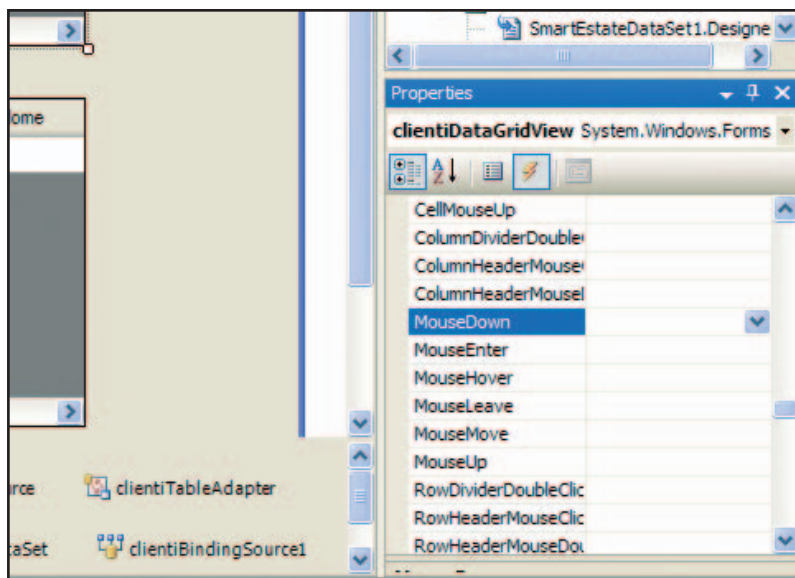
Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

COME FARE DRAG'N'DROP DA UNA DATAGRIDVIEW AD UN'ALTRA

UN TRUCCO UTILE PER COPIARE I DATI DA UNA TABELLA AD UN'ALTRA IN MODO SEMPLICE E IMMEDIATO. VEDIAMO COME

1 Prima di tutto una connessione a un database. Facciamolo con il *Wizard* di Visual Studio cliccando sul menu *"Data/Add new data source"*.

2 Nella form che segue scegliamo *"DataBase"* e poi *next*

3 Scegliamo *"New Connection"* per stabilire la fonte da cui prelevare i dati

4 Nella form che segue selezioniamo il tipo di database a cui fare il binding e ovviamente la fonte dei dati

5 Confermiamo la scelta del database e proseguiamo cliccando su *Next*

6 Salviamo la stringa di configurazione e andiamo avanti

7 Scegliamo le tabelle da importare e terminiamo il Wizard.

8 Trasciniamo dal dataset sulla form la tabella che ci interessa e creiamone una che disponga di almeno due campi uguali a quelli della tabella sorgente

9 Nella griglia di destinazione settiamo a *true* la property *"AllowDrop"*

10 Nella griglia sorgente clicchiamo due volte sull'evento *"MouseDown"* per generarne il codice di gestione

11 Il codice da inserire è il seguente

```
private void clientiDataGridView_MouseDown(
                        object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Right)
   {
      DataGridView.HitTestInfo info =
                  clientiDataGridView.HitTest(e.X, e.Y);
      if (info.RowIndex >= 0)
      {
         DataRowView row = (DataRowView)
             clientiDataGridView.Rows[info.RowIndex]
                                        .DataBoundItem;
         if (row != null)
            clientiDataGridView.DoDragDrop(
                  row, DragDropEffects.Copy); } }
}
```

12 Nella datagrid destinazione gestiamo prima l'evento *DragEnter*, generando il template di gestione, cliccando due volte sulla voce che lo rappresenta. Il codice da inserire è il seguente:

```
private void clientiDataGridView1_DragEnter(
                        object sender, DragEventArgs e)
        { e.Effect = DragDropEffects.Copy; }
```

13 Infine gestiamo l'evento *DragDrop* che al rilascio del pulsante del mouse copia fisicamente i dati selezionati

```
private void clientiDataGridView1_DragDrop(
                        object sender, DragEventArgs e)
        {
            if (e.Data.GetDataPresent(
                        typeof(DataRowView)))
            { DataRowView row = (DataRowView)e.
```

```
                    Data.GetData(typeof(DataRowView));
               clientiDataGridView1.Rows.Add(
                    row["Nome"].ToString(),
                    row["Cognome"].ToString()); }
    }
```

COME FUNZIONA?

Prima di ogni viene impostata a true la property AllowDrop che rende attiva la possibilità di gestire il drag & drop sulle due grid. Alla pressione del tasto destro del mouse sulla datagrid sorgente viene invocato l'evento MouseDown che controlla che la riga selezionata contenga almeno qualche dato. Se la riga non è nulla la copia in un buffer. Quando il bottone del mouse viene rilasciato vengono aggiunti i dati alla griglia di destinazione.

FACCIAMOLO IN VISUAL BASIC

1 I passi da uno a dieci rimangono identici a quelli utilizzati per C#. Il codice di gestione dell'evento *MouseDown* diventa

```
Private Sub ClientiDataGridView_MouseDown(
        ByVal sender As System.Object, ByVal e As
        System.Windows.Forms.MouseEventArgs)
        Handles ClientiDataGridView.MouseDown
   If e.Button=Windows.Forms.MouseButtons.Right Then
      Dim info As DataGridView.HitTestInfo =
            ClientiDataGridView.HitTest(e.X, e.Y)
      If (info.RowIndex >= 0) Then
         Dim bound As Integer = info.RowIndex
         Dim tmp As Object = ClientiDataGridView.Rows(
               bound).DataBoundItem
         Dim row As DataRowView = CType(
               tmp, DataRowView)
         If Not row Is Nothing Then
            ClientiDataGridView.DoDragDrop(
               row, DragDropEffects.Copy)
         End If
      End If
End If
```

2 Il codice di gestione dell'evento *DragEnter* diventa

```
Private Sub DataGridView1_DragEnter(ByVal sender
        As System.Object, ByVal e As
        System.Windows.Forms.DragEventArgs)
        Handles DataGridView1.DragEnter
   e.Effect = DragDropEffects.Copy
End Sub
```

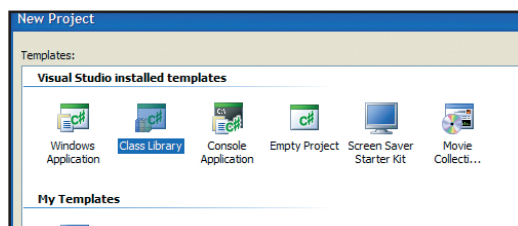
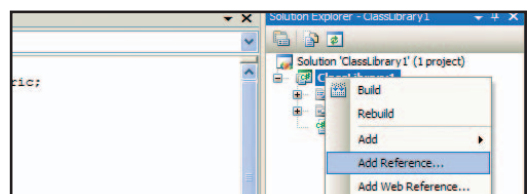
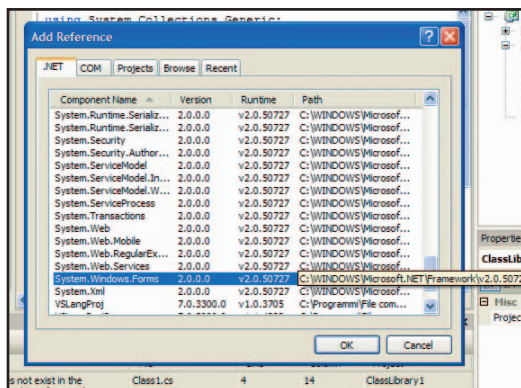
3 Infine il codice di gestione dell'evento *Drag-Drop* si può riscrivere come segue

```
Private Sub DataGridView1_DragDrop(ByVal sender
        As System.Object, ByVal e As
        System.Windows.Forms.DragEventArgs)
        Handles DataGridView1.DragDrop
   If (e.Data.GetDataPresent(GetType(
         DataRowView))) Then
      Dim tmp As Object = e.Data.GetData(
            GetType(DataRowView))
      Dim row As DataRowView = CType(
            tmp, DataRowView)
      DataGridView1.Rows.Add(row("Nome")
         .ToString(), row("Cognome").ToString())
   End If
End Sub
```

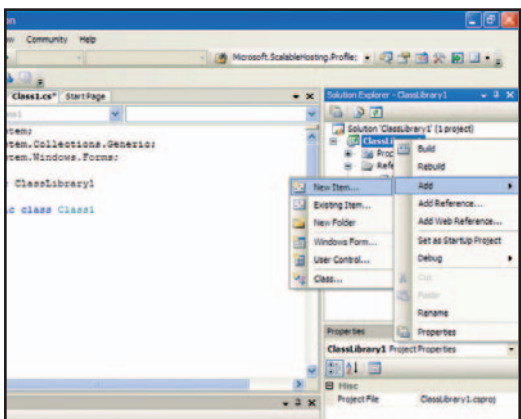
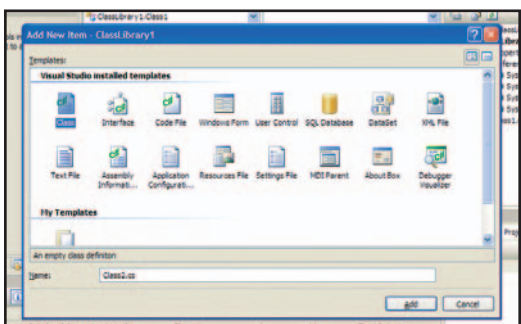
COME POSSO AGGIUNGERE UN'IMMAGINE AGLI ELEMENTI DI UNA COMBOBOX?

CREEREMO UNA NUOVA CLASSE DERIVATA DALL'ORIGINALE E SFRUTTEREMO LA PROPRIETÀ DRAWMODE PER DETERMINARE IL DISEGNO DELLA COMBO

C#

1 Clicchiamo su *"File/New Project"* e poi scegliamo class library

2 Clicchiamo con il tasto destro del mouse sul *Solution explorer* e di seguito alla voce *"Add /Reference"*

3 Nella dialog box che compare selezioniamo System.Windows.Form e System.Drawing

4 In alto nel codice aggiungiamo gli import corretti

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel;
using System.Drawing;
```

5 Clicchiamo con il tasto destro del mouse nel solution explorer alla voce add/new item

6 Nella dialog box che compare selezioniamo class

7 Nella nuova classe che abbiamo così definito aggiungiamo il seguente codice

```
public class ImageComboItem : Component
{
   private Object m_object;
   private Image m_Image;
   public ImageComboItem(object item,Image img)
   {
      m_object=item;
      m_Image=img;
   }

   [TypeConverter(typeof(StringConverter))]
   public Object Item
   {
      get{return m_object;}
      set{m_object = value;}
    }

    public Image Image
   {
      get{return m_Image;}
      set{m_Image = value;}
   }
   public override string ToString()
   {
      if (m_object == null)
         return String.Empty;
      else
         return m_object.ToString();
    }

}
```

8 Infine nella classe primaria aggiungiamo il seguente codice e compiliamo il tutto tramite il menu build

```
public class ImageCombo :
                        System.Windows.Forms.ComboBox
{

   private System.ComponentModel.Container
```

COME PROVARE LA NUOVA CLASSE

Aprite un nuovo progetto e cliccate con il tasto destro del mouse su un punto vuoto della *tabsheet*. Selezionate dal menu a tendina che comparirà la voce *"Add Tab"*. Nella dialog box che segue selezionate *"Browse"* e cercate nel vostro hard disk la dll che avete creato in precedenza.

La nuova dll comparirà nei tool con l'icona di una ruota dentata.

Trascinatela sulla form e in relazione all'evento *onload* della form aggiungete il seguente codice

```
private void Form1_Load(
             object sender, EventArgs e)
   {
      imageCombo1.Items.Add(
             new ImageComboItem(
             "item1", Image.FromFile(
             @"c:\immagine.bmp")
             ));
   }
```

non dimenticate di aggiungere la direttiva using *ClassLibrary1*; all'inizio del vostro codice.

```
                                components = null;
public ImageCombo()
{ base.DrawMode = DrawMode.OwnerDrawVariable;}
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if(components != null)
        { components.Dispose();}
        if (this.Items != null)
        {
            foreach (Object o in this.Items)
            {
                if (o is ImageComboItem)
                    ((ImageComboItem)o).Dispose();
            }
        }
    }
    base.Dispose( disposing );
}
private int currentIndex = -1;

protected override void OnDrawItem(
                            DrawItemEventArgs e)
{
    if (e.Index == -1 || e.Index
                        > this.Items.Count - 1)
    return;
    e.DrawBackground();
    Rectangle imageRect = new Rectangle(
        e.Bounds.X, e.Bounds.Y, .Bounds.Height,
                            e.Bounds.Height);
    RectangleF textRectF = RectangleF.FromLTRB(
            imageRect.Right + 2, e.Bounds.Top,
            e.Bounds.Right, e.Bounds.Bottom);

    if (Items[e.Index] is ImageComboItem )
    {
        ImageComboItem Item =
            (ImageComboItem)Items[e.Index];
        if (Item.Image != null)
            e.Graphics.DrawImage(Item.Image,
                                    imageRect);
    }
    SolidBrush TextBrush = new
                    SolidBrush(this.ForeColor);
    if ((e.State & DrawItemState.Selected)
                    == DrawItemState.Selected)
    TextBrush.Color = SystemColors.HighlightText;
        StringFormat sf = new StringFormat(
                    StringFormatFlags.NoWrap);
        sf.LineAlignment = StringAlignment.Center;
        sf.Trimming =
            StringTrimming.EllipsisCharacter;
        e.Graphics.DrawString(
            Items[e.Index].ToString(), this.Font,
                    TextBrush, textRectF, sf);
        TextBrush.Dispose();
    }
    protected override void
        OnSelectedIndexChanged(EventArgs e)
    {
        if (this.SelectedIndex != this.currentIndex)
        {
            this.currentIndex = this.SelectedIndex;
            base.RefreshItem(this.SelectedIndex);
        }
        else
        {
            base.OnSelectedIndexChanged (e);
        }
    }
}
```

COME FUNZIONA

Per creare una ComboBox, che oltre alle stringhe visualizzi un'immagine associata ad esse, dobbiamo creare una nuova classe, derivandola dalla ComboBox standard. Come modalità di disegno utilizziamo il valore DrawMode.OwnerDrawVariable, impostando la proprietà DrawMode, e ciò implica che saremo noi a dover gestire il disegno degli elementi della ComboBox, che saranno rappresentati dalla classe ImageComboItem, il cui scopo è incapsulare il testo e l'immagine di ogni item. La classe ImageCombo dovrà fornire un override del metodo OnDrawItem per disegnare l'immagine di ogni Item, e di seguito la relativa stringa.

COME POSSO CONVERTIRE HTML IN TESTO

Vogliamo scrivere una funzione che converta una pagina HTML in testo puro, eliminando tutti i Tags HTML. Per farlo possiamo utilizzare lo spazio dei nomi *System.Text.RegularExpressions*. All'interno di *System.Text.RegularExpressions*, sono racchiuse tutte le classi che consentono di accedere al modulo delle espressioni regolari di VB.NET 2003. In particolare utilizziamo la classe Regex che rappresenta un'espressione regolare non modificabile. Tra i suoi metodi ci viene in aiuto il metodo *Replace*. Con il metodo *Replace* possiamo sostituire tutte le ricorrenze delle corrispondenze definite dall'espressione regolare con una stringa di sostituzione, a partire dal primo carattere nella stringa di input.

CHE COSA VUOL DIRE CASTING?

SI TRATTA DI UNA TECNICA CHE CONSENTE DI CONVERTIRE UN TIPO DI DATO IN UN ALTRO, DA ADOTTARE CON MODALITÀ PARTICOLARI PER OGNI LINGUAGGIO. VEDIAMO COME

C#

VISUAL BASIC

FACCIAMOLO IN C#

Una delle operazioni più comuni da effettuare in un qualsiasi programma, è la conversione di un dato da un tipo ad un altro. C# è un linguaggio *strongly-typed*, cioè fortemente tipizzato, dunque quando due tipi non sono compatibili deve essere esplicitamente indicata la conversione che si vuol eseguire.
Ad esempio si parla di conversione implicita quando si vuol convertire un dato di tipo byte in *int*:

```
byte b=5;
int i=b;
```

nel caso in cui non sia possibile una conversione implicita, si ricorre all'operatore di cast *()*, che si utilizza preponendolo al dato da convertire, e inserendo fra le parentesi il tipo di destinazione.

```
int i=13;
byte b1=i; //errore Cannot implicitly convert type
                                    'int' to 'byte'
byte b2=(byte)i; //cast di int verso byte
```

In alcuni casi l'operazione di cast può provocare perdita di informazione, quindi è necessario sapere esattamente ciò che si vuole ottenere. Ad esempio convertendo un double in int per mezzo di un casting si avrebbe:

```
double d=1.432;
int i=(int)d;
Console.WriteLine(i); //stampa 1
```

Si è persa così la precisione del tipo double.
In altri casi l'operazione di cast, seppur possibile a tempo di compilazione, potrebbe provocare un eccezione *InvalidCastException:*

```
object obj="string";
...
int i=(int)obj;
```

Il cast di *object* verso *int* è permesso dal compilatore, in quanto esso non può sapere a compile-time che tipo di dato è contenuto in un *object*. Durante l'esecuzione del programma però si verificherebbe un'eccezione, in quanto si tenterebbe di convertire una stringa in un numero intero.
Un'altra possibilità, applicabile solo ai tipi riferimento, è l'utilizzo dell'operatore *as*.

```
Object obj = new TextBox();
Button btn = obj as Button;
```

Il vantaggio nell'utilizzare l'operatore *as* in questo caso, è che esso restituirà *null* se la conversione non è possibile, senza lanciare eccezioni.

FACCIAMOLO IN VB.NET

In visual basic.NET le conversioni sono regolate per mezzo dell'istruzione *Option Strict* posta all'inizio di un file di codice.
Se scrivete:

```
Option Strict On
```

allora si ha il controllo *strict* sui tipi e non saranno possibili conversioni implicite fra tipi incompatibili. Ad esempio

```
Dim i as Integer
Dim d as Double
d=1.234
i=d
```

darà errore a compile-time in quanto non esiste una conversione implicita da double a integer.
Lo stesso codice con l'*Option Strict Off* convertirebbe correttamente il *double* in *integer*, naturalmente perdendo le cifre decimali, e restituendo 1.
Nei casi in cui la conversione implicita non sia possibile è necessario ricorrere al casting esplicito, che si effettua per mezzo della funzione *CType*, che prende come primo argomento il dato da convertire e per secondo il tipo destinazione:

```
i=CType(d,Integer)
```

Se la conversione non è possibile, viene generata un'eccezione *InvalidCastException*.
Esistono poi funzioni specializzate per un particolare tipo destinazione che prendono come parametro solo il dato da convertire,

ad esempio *CBool* per convertire in boolean il parametro, *CStr* per la conversione in *String* e così via:

```
dim boolVar as Boolean
dim str as String
boolVar=Cbool(i)
str=CStr(i)
```

Un'altra funzione utilizzabile per la conversione esplicita da e verso il tipo *object* è la *DirectCast*, che si usa in modo analogo alla *CType*.
In questo caso però perché sia permessa una conversione fra due tipi, è richiesta una relazione di ereditarietà o implementazione fra di essi.
Ad esempio:

```
Dim q As Object = 9.456
i = CType(q, Integer) ' ok
Dim j As Integer = DirectCast(q, Integer) 'fallisce
```

La *CType* va a buon fine come già visto, perché esiste una possibile conversione da *Double* a *Integer*, mentre la *DirectCast* fallisce generando una *InvalidCastException*.

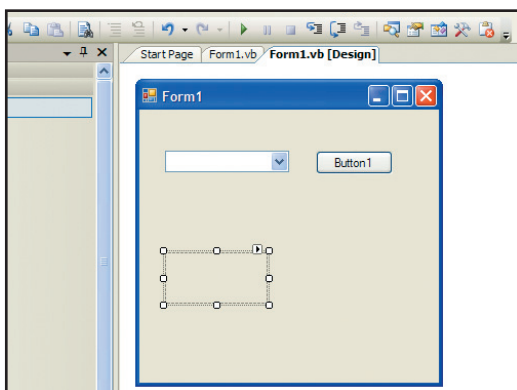
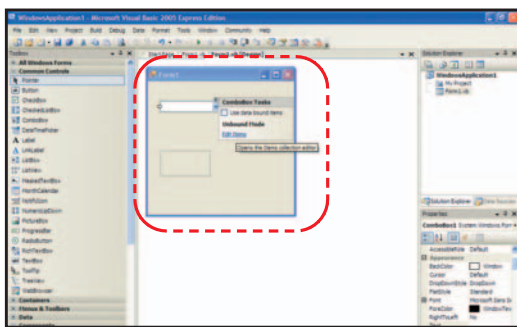
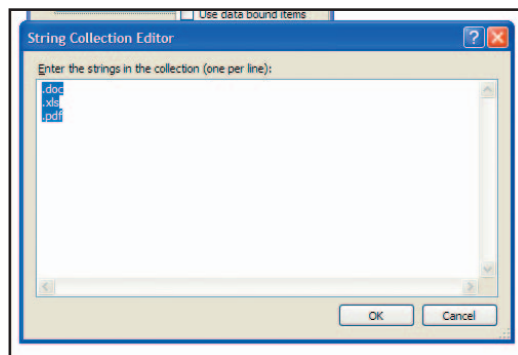
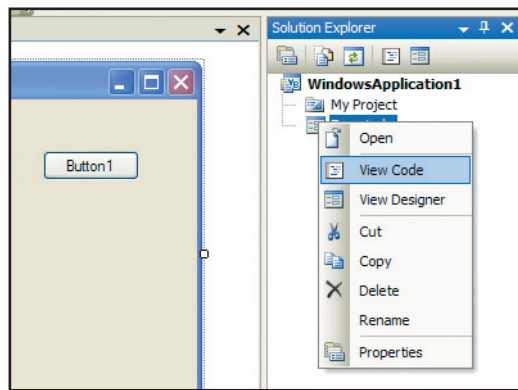
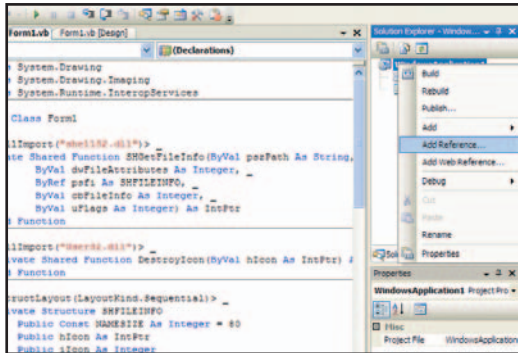
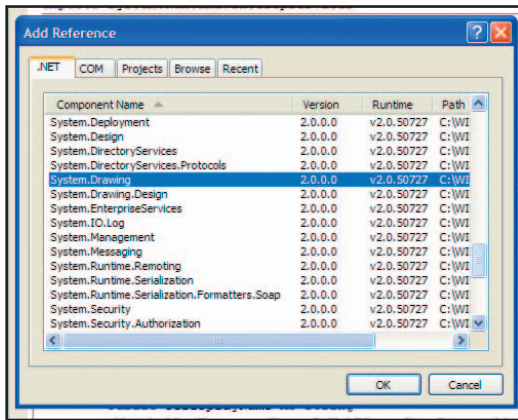
```
Dim f As New Form
Dim c As Control
c = DirectCast(f,  Control)  'ok
```

In questo secondo caso la *DirectCast* va a buon fine, dato che Form deriva da *Control*. Analoga alla *DirectCast* è la *TryCast* che però invece di generare una eventuale eccezione restituisce il valore *Nothing*.

COME POSSO VISUALIZZARE L'ICONA ASSOCIATA AD UN FILE?

UTILIZZEREMO IN MODO SEMPLICE ALCUNE DLL DI SISTEMA PER RAGGIUNGERE LO SCOPO, VEDREMO CHE È PIÙ FACILE DI QUELLO CHE NORMALMENTE SI PENSA

VISUAL BASIC.NET

1 Creiamo una nuova windows application e nella form principale trasciniamo un combobox, un componente PictureBox, un bottone

2 Utilizziamo lo smarttag associato al combobox per inserire le voci relative alle estensioni di cui vogliamo conoscere l'icona

3 Nella dialog box successiva aggiungiamo alcune estensioni che ci interessano, una per riga

4 Agiamo con il tasto destro del mouse sul solution explorer selezionando la form e dal menu contestuale scegliamo "view code"

5 Agendo ancora una volta sul solution explorer con il tasto destro del mouse, scegliamo questa volta “Add Reference”

6 Dalla dialog box successiva scegliamo “System .drawing” e infine clicchiamo su OK

7 Dichiariamo le funzioni di sistema che andremo ad utilizzare e le relative DLL che le contengono

```
<DllImport("shell32.dll")> _
Private Shared Function SHGetFileInfo(ByVal
                                    pszPath As String, _
                      ByVal dwFileAttributes As Integer, _
                            ByRef psfi As SHFILEINFO, _
                            ByVal cbFileInfo As Integer, _
                        ByVal uFlags As Integer) As IntPtr
  End Function

  <DllImport("User32.dll")> _
  Private Shared Function DestroyIcon(ByVal hIcon
                                    As IntPtr) As Integer
  End Function

  <StructLayout(LayoutKind.Sequential)> _
  Private Structure SHFILEINFO
     Public Const NAMESIZE As Integer = 80
     Public hIcon As IntPtr
     Public iIcon As Integer
     Public dwAttributes As Integer
     <MarshalAs(UnmanagedType.ByValTStr,
                                     SizeConst:=260)> _
      Public szDisplayName As String
      <MarshalAs(UnmanagedType.ByValTStr,
                                      SizeConst:=80)> _
      Public szTypeName As String
   End Structure
```

8 Aggiungiamo qualche costante che ci tornerà utile nel resto del procedimento

```
Private Const FILE_ATTRIBUTE_NORMAL = &H80
   Private Const FILE_ATTRIBUTE_DIRECTORY = &H10
   Private Const SHGFI_ICON = &H100
' get icon
   Private Const SHGFI_OPENICON = &H2
' get open icon
   Private Const SHGFI_USEFILEATTRIBUTES = &H10
' use passed dwFileAttribute
```

9 Infine scriviamo il codice relative alla funzione che estrarrà le icone utilizzando le dll di cui sopra

```
   Public Enum IcoTypeEnum
      File = FILE_ATTRIBUTE_NORMAL
      Directory = FILE_ATTRIBUTE_DIRECTORY
   End Enum

   Public Enum IcoSizeEnum ' grandezza icona
      Small = &H0 ' 16x16
      Large = &H1 ' 32x32
   End Enum

   Public Function GetFileIcon(ByVal ext As String,
               ByVal IcoSize As IcoSizeEnum) As Icon
      Dim hImg As IntPtr  'handle all'image list di
                                            sistema.
      Dim shinfo As New SHFILEINFO
      Dim flags As Integer =
        SHGFI_USEFILEATTRIBUTES Or SHGFI_ICON
      flags += IcoSize
      If Not ext.StartsWith(".") Then ext = "." & ext
      hImg = SHGetFileInfo(ext,
FILE_ATTRIBUTE_NORMAL, shinfo, _
                              Marshal.SizeOf(shinfo),
                              flags) ' valorizza l'handle
      Dim myIcon As System.Drawing.Icon
      myIcon = System.Drawing.Icon.FromHandle(
                                   shinfo.hIcon).Clon
      DestroyIcon(shinfo.hIcon)
      Return myIcon
   End Function
```

10 Torniamo in modalità disegno e ciccando due volte sul bottone otteniamo il template per l’evento *OnClick*, il relativo codice è il seguente

```
Private Sub Button1_Click(ByVal sender As
```

```
        System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim ico As Icon = GetFileIcon(
                ComboBox1.SelectedItem.ToString(),
                        IcoSizeEnum.Small)
        Dim bmp As System.Drawing.Bitmap =
                        ico.ToBitmap()
        PictureBox1.Image = bmp
    End Sub0
```

COME FUNZIONA?

Il cuore del sistema è la funzione SHGetFileInfo che è dichiarata nella DLL Shell32.dll. Per potere utilizzare questa funzione la ridichiariamo all'inizio del codice.
La SHGetFileInfo fa uso di alcune strutture come la SHFILEINFO che dichiariamo sempre all'interno del nostro codice. L'unica funzione che andiamo realmente a riscrivere secondo le nostre necessità è la GetFileIcon che riceve in input una stringa che rappresenta l'estensione e sfrutta la SHGetFileInfo per ritornare un oggetto di tipo icon. Nell'evento OnClick del bottone non facciamo altro che passare i valori corretti alla GetFileIcon e mostrare tutto nella picturebox

GLI IMPORT DA UTILIZZARE

All'inizio del codice è necessario inserire le seguenti linee

```
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices
```

COME POSSO SAPERE SE UN NUMERO È PARI O DISPARI?

Nella rappresentazione binaria, tutti i numeri pari hanno il bit meno significativo (quello più a destra) pari a zero (0) mentre i numeri dispari pari ad uno(1) In VB.Net 2003 si può applicare l'operatore And ai valori numerici. In questo caso l'operatore And consente di eseguire un confronto bit per bit dei bit che occupano la stessa posizione in due espressioni numeriche diverse. Per i nostri scopi possiamo mettere in And il numero da testare, con il valore uno (1)

```
Private Function NumeroDispari(
        ByVal NumeroDaTestare As Long)
                        As Boolean
    NumeroDispari = NumeroDaTestare
                        And 1
End Function
```

QUALI SONO LE FUNZIONI PER MANIPOLARE STRINGHE IN VB.NET 2003?

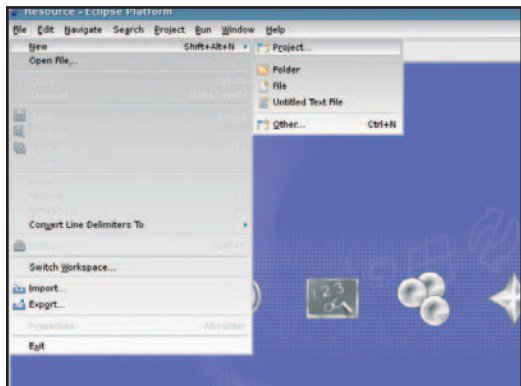
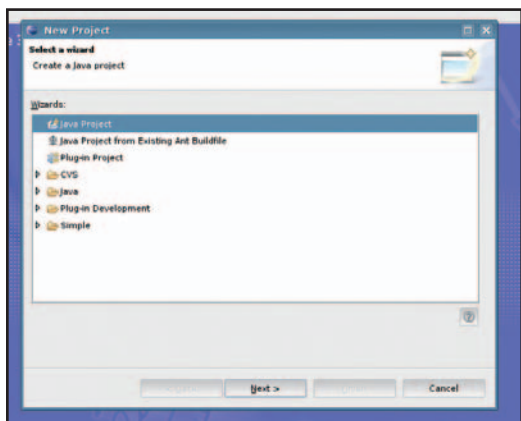
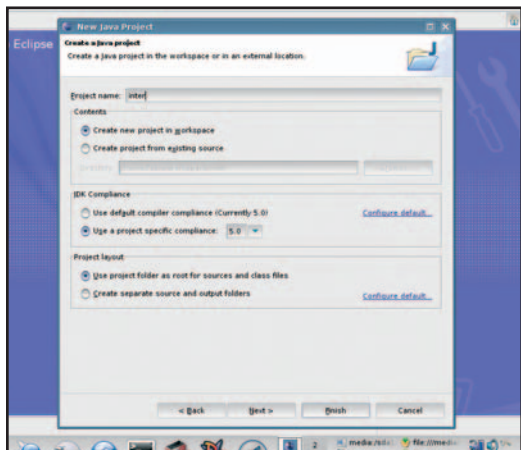
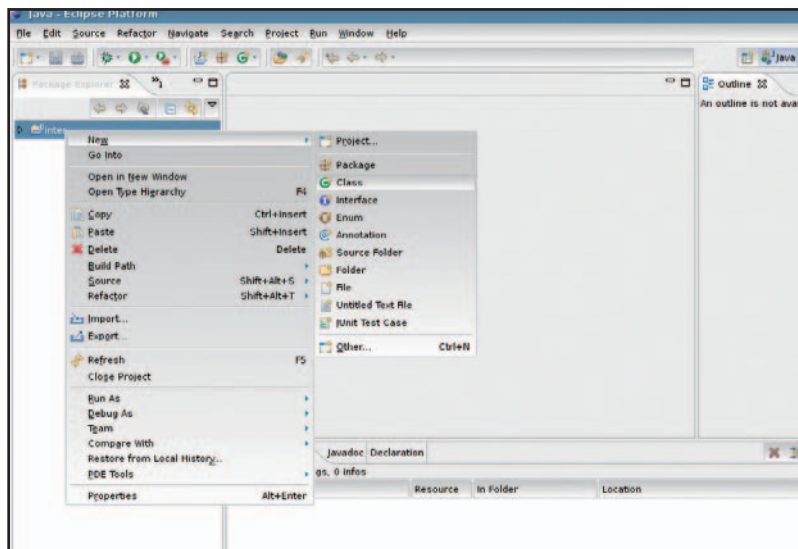
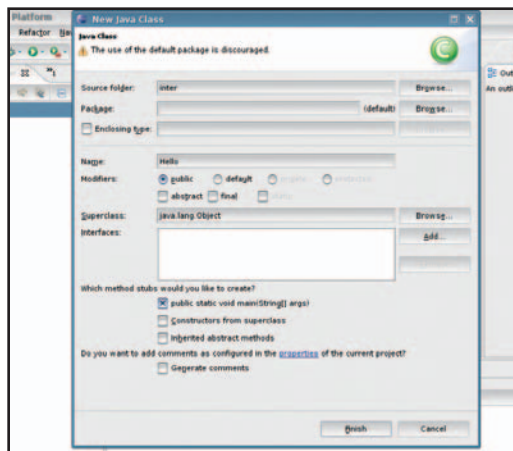
- **Chars** - restituisce il carattere che corrisponde alla posizione passata come argomento.
- **IndexOf** - restituisce la posizione della prima occorrenza di un carattere o stringa a partire dalla posizione indicata.
- **LastIndexOf** - funziona in modo opposto al metodo *IndexOf*, restituisce l'indice dell'ultima occorrenza di un carattere o stringa.
- **Substring** - restituisce una sottostringa della stringa passata come argomento, a partire da una posizione fino al numero di caratteri specificato.
- **StartsWith** - verifica se la stringa di test contiene come stringa iniziale quella indicata come parametro. Restituisce *True* o *False*.
- **EndsWith** - funziona in modo opposto controllando la stringa finale. Restituisce *True* o *False*.
- **Insert** - inserisce, all'interno della stringa esistente, una sottostringa a partire da un indice di posizione.
- **Remove** - elimina una sottostringa da una stringa esistente a partire da una posizione per uno specifico numero di caratteri.
- **Lenght** - restituisce il numero di caratteri che compongono la stringa.

```
Dim sTest As String = "IoProgrammo.it"
MessageBox.Show(sTest.Chars(2))
                        'Visualizza P
MessageBox.Show(sTest.IndexOf("m", 0))
                        'Visualizza 8
MessageBox.Show(sTest.LastIndexOf("m"))
                        'Visualizza 9
MessageBox.Show(sTest.Substring(2, 9))
                'Visualizza Programmo
MessageBox.Show(sTest.StartsWith("Io"))
                    'Visualizza True
MessageBox.Show(sTest.EndsWith(".it")) '
                    Visualizza True
MessageBox.Show(sTest.Insert(0, "www.")) '
            Visualizza www.IoProgrammo.it
MessageBox.Show(sTest.Remove(11, 3)) '
                Visualizza IoProgrammo
MessageBox.Show(sTest.Length) '
                        Visualizza 14
```

COME POSSO REALIZZARE APPLICAZIONI MULTILINGUE CON JAVA?

POSSIAMO UTILIZZARE ALCUNE DELLE CARATTERISTICHE PECULIARI DEL LINGUAGGIO. VEDIAMO UN ESEMPIO CON ECLIPSE

JAVA

1 Apriamo eclipse e scegliamo *File/New Project*

2 Dalla dialog box che segue scegliamo *"Java Project"*

3 Diamo un nome significativo al progetto e lasciamo le altre opzioni come ci vengono proposte da eclipse

4 Aggiungiamo una nuova classe al progetto agendo con il tasto destro del mouse sul nome del progetto e scegliendo poi *"New /Class"* proposte da eclipse

5 Diamo un nome significativo alla classe e clicchiamo su finish. Facciamo generare automaticamente il main utilizzando l'apposito flag

6 Aggiungiamo in testa al codice gli import che ci serviranno

```
import java.text.MessageFormat;

import java.util.Locale;

import java.util.ResourceBundle;
```

7 Modifichiamo il main come segue

```
public static void main(String[] args)
{
  // TODO Auto-generated method stub
  if (args != null && args.length > 0)
 {
   // Saluta in italiano ...
   Hello itHello = new Hello(Locale.ITALIAN);
   System.out.println(itHello.greet(args[0]));

   // ... e poi in inglese
   Hello enHello = new Hello(Locale.ENGLISH);
   System.out.println(enHello.greet(args[0]));
 }
 else
 {
    System.out.println("Usage java
                                    Hello [your name]");
    System.exit(-1);
 }

}
```

8 Aggiungiamo qualche variabile privata e creiamo il costruttore della classe

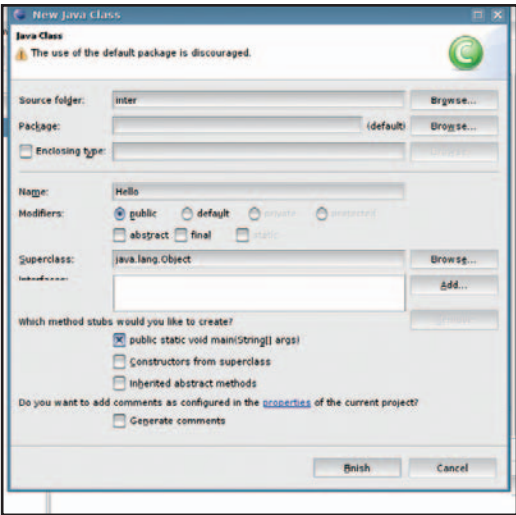
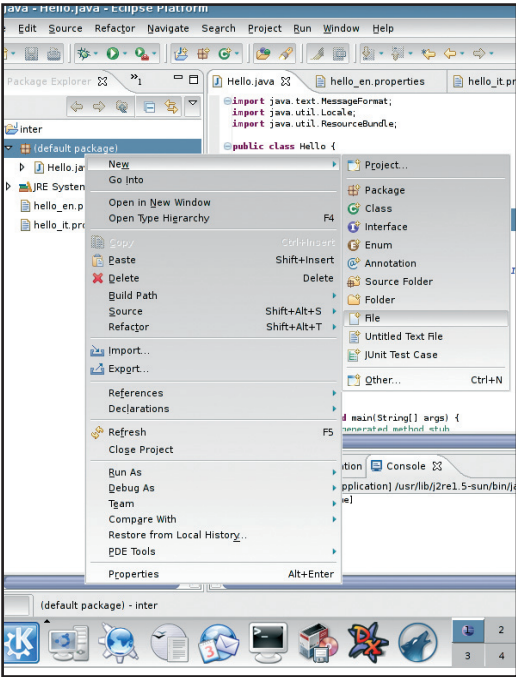
```
private ResourceBundle bundle;
          private static final String FILE_PREFIX =
                                                  "hello";
          private static final String GREET_KEY    =
                                                  "greet";

          public Hello(Locale locale)
          {
               this.bundle =
                              ResourceBundle.getBundle(
                              Hello.FILE_PREFIX, locale);
  }
```

9 Infine definiamo il metodo string_greet

```
public String greet(String name)
{
 // estrae la stringa ...
 String msg = this.bundle.getString(GREET_KEY);
 StringBuffer msg_f = new StringBuffer();

 // ... e la formatta
 MessageFormat mf = new MessageFormat(msg);
 mf.format(new String[]{ name }, msg_f, null);
  return msg_f.toString();
}
```

10 Aggiungiamo due file rispettivamente *hello_en.properties*, e *hello_it.properties*, agendo con il pulsante destro sul nome del progetto e poi su *New/File*

11 I file in questione devono contenere la definizione delle stringhe di linguaggio come segue:

12 Compiliamo il tutto e lanciamo l'applicazione con java

```
Hello [Nome]
```

ci verrà restituito il saluto in due lingue

COME FUNZIONA?

Tutto sta nel richiedere il giusto file di internazionalizzazione, la stringa viene estratta dal metodo *String_Greet* che restituisce la corretta localizzazione.

COME POSSO AVERE INFORMAZIONI SULLE CLASSI CARICATE?

UN ESEMPIO PRATICO PER IMPARARE COME VENGONO RICHIAMATE LE CLASSI ALL'INTERNO DI JAVA E COME VIENE GESTITA LA MEMORIA

In questo tip illustreremo come tracciare l'attività di due componenti di primaria importanza all'interno della Java Virtual Machine: ClassLoader e Garbage Collector. Il primo è responsabile del caricamento delle classi richiamate durante il ciclo di vita dell'applicazione, mentre il secondo si prende cura di liberare, in maniera trasparente agli occhi dello sviluppatore, gli oggetti non referenziati all'interno della heap memory. Per capire meglio tali attività useremo un'applicazione di esempio che stampa un valore double casuale compreso tra 0.0 e 1.0. Nel metodo main è stata introdotta una chiamata esplicita al metodo System.gc() in modo da simulare un'esecuzione del Garbage Collector.

```
/**
 * Classe di esempio per la stampa di un numero
 * (double) casuale

 */
public class RandomPrinter {

    /**
     * Stampa un numero random a video
     */
    public void print() {
        System.out.println("Random: " +
                                    Math.random());
    }

    /**

     * Entry point dell'applicazione

     */
    public static void main(String[] agrs) {
        System.out.println("START");
        RandomPrinter rp = new RandomPrinter();
        rp.print();
        System.gc();
        System.out.println("END");
    }
}
```

Eseguendo la nostra applicazione di esempio con il comando “java” e l'opzione -verbose:class, avremo un output molto simile al seguente (la formattazione potrebbe dipendere dall'implementazione della JVM):

```
>java -verbose:class RandomPrinter
[Opened /usr/java/jdk1.5.0_01/jre/lib/rt.jar]
[Opened /usr/java/jdk1.5.0_01/jre/lib/jsse.jar]
[Opened /usr/java/jdk1.5.0_01/jre/lib/jce.jar]
[Opened
        /usr/java/jdk1.5.0_01/jre/lib/charsets.jar]
[Loaded java.lang.Object from shared objects
                                            file]
[Loaded java.io.Serializable from shared objects
                                            file]

.... seguono altri 330 caricamenti

[Loaded java.util.regex.Pattern$TreeInfo from
                               shared objects file]
Random: 0.6229570234648002
[Loaded java.lang.Enum from shared objects file]
END
[Loaded java.lang.Shutdown from shared objects
                                            file]
[Loaded java.lang.Shutdown$Lock from shared
                                    objects file]
```

Nonostante la palese semplicità della classe monitorata, il ClassLoader carica oltre 300 classi, la cui origine risiede nei quattro jar files aperti all'inizio dell'esecuzione.
Se nel comando eseguito precedentemente si cambia l'opzione class con gc vengono invece tracciate le esecuzioni del Garbage Collector:

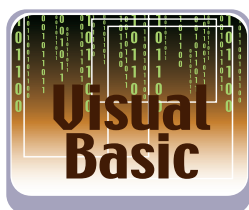
```
>java -verbose:gc RandomPrinter
START
Random: 0.9247276620083805
[Full GC 196K->103K(1984K), 0.0228140 secs]
END
```

- **Full GC**: indica che il processo del GC ha attraversato l'intera heap memory.

Il significato delle stampe del GC è abbastanza semplice:

- **196K**: era la dimensione della heap prima dell'esecuzione del GC.
- **103K**: è la dimensione della heap dopo dell'esecuzione del GC.
- **1984K**: è la memoria disponibile (memoria totale – memoria occupata dagli oggetti “vivi”).
- **0.0228140 secs**: è il tempo impiegato dal GC.

SQL SERVER EXPRESS 2005 E LE STORED PROCEDURE

DESCRIVEREMO COME REALIZZARE UN'APPLICAZIONE CLIENT-SERVER MULTIMEDIALE CHE UTILIZZA MODULI DI CLASSE, STORED PROCEDURE E UN DATABASE MICROSOFT SQL SERVER EXPRESS. OVVIAMENTE LO FAREMO UTILIZZANDO VISUAL BASIC 6.0

Nel precedente appuntamento abbiamo presentato *Microsoft SQL Server 2005 Express Edition* ed alcuni elementi ad esso collegati come *SQL Server Management Studio Express* - un *free database Manager* - e *SQL Native Client* la nuova tecnologia per l'accesso ai dati. Come esempio abbiamo introdotto un'applicazione *Client/Server* che gestisce una *mediateca*, cioè un archivio con una serie di informazioni sui dati contenuti in dei *supporti multimediali* (*DVD*, *CD* ecc.). Il *database* dell'applicazione, nominato *DVD*, tra l'altro, include la tabella *Supporto* con i dati principali dei supporti ottici. Nell'esempio l'interazione tra l'applicazione e il *database* è gestita tramite delle *Query SQL* invocati da oggetti *ADODB*. In questo appuntamento continuiamo la trattazione su *SQL Express* e presentiamo altre modalità d'interazione con i database quali *Stored Procedure*, *T-SQL Debugger* e *Tool Data View*. Per quanto riguarda l'applicazione d'esempio continueremo con la *gestione* della *mediateca*, introducendo nuove tabelle e nuove funzionalità. In realtà, manteniamo l'idea fondamentale, cioè quella di archiviare i dati dei supporti ottici, ma modifichiamo sia gli elementi del database che quelli dell'interfaccia e presentiamo una tecnica d'interazione tra *Visual Basic* e *DBMS* basata sui moduli di classe e *Stored Procedure.*

ADO E STORED PROCEDURE

Le *Stored Procedure (SP)* sono degli *script* collocati nel *DBMS*. Esse sono composte da istruzioni *SQL* e istruzioni per il controllo di flusso. Una *SP* è identificata attraverso un nome ed è elaborata come una singola unità. Ai fini della programmazione una *SP* può essere vista come un metodo, infatti può ricevere e restituire parametri. Le *SP* sono compilate solo alla prima esecuzione. La sintassi, non completa per la creazione di una *Stored Procedure* è la seguente.

```
Create Procedure NomeProcedura
 ( @nomeparametroIn datatype [= valore di default],
@nomeparametroInOut datatype [= valore di default]
                                          OUTPUT )
As Istruzioni_Transact-SQL [ ...n ]
```

I parametri possono essere di *Input* di *Output* o di *InputOutput*. Ad ogni parametro è associato un tipo (*Datatype*) e un valore (opzionale) di *Default*. Il nome del parametro è preceduto dal carattere @. Le istruzioni della *SP*, sono specificate dopo l'elenco dei parametri. Come vedremo negli esempi le SP sono invocate utilizzando gli oggetti ADODB. In particolare sono utilizzati alcuni elementi dell'*ADO Connection*. Per specificare la *SP*, da eseguire, si utilizza la proprietà *CommandText*; la *CommandType*, invece è utilizzata per indicare il tipo di *Command* che deve essere utilizzato (cioè *Command* di tipo *Stored Procedure*). Negli esempi utilizzeremo anche i metodi: *CreateParameter* ed *Execute*. Il primo crea un nuovo parametro con un *Nome* (valore opzionale), un *tipo*, una *direzione* (per esempio *adParamInput* oppure *adParamOutput*), una *dimensione* (dipende dal tipo) e un *valore*. Il metodo *Execute*, invece, avvia l'esecuzione della *Stored Procedure*. Ricordiamo che un *Command* è un oggetto che può essere usato per eseguire una *query* o appunto una *Stored Procedure* e che l'oggetto *Parameters* rappresenta l'insieme dei parametri (insieme di oggetti *Parameter*) dell'oggetto *Command*.

CREARE ED ESEGUIRE UNA SEMPLICE STORED PROCEDURE

Descriviamo come creare, eseguire e controllare la *Stored Procedure* che restituisce il valore massimo contenuto nel campo *SupportoID* della tabella *Supporto* (descritta nella **Tabella 1**).

1 Con SQL Server Management Studio create il database DVD con almeno la tabella Supporto. Se non avete voglia di scrivere i nomi delle tabelle e dei relativi campi, per creare il database, potete utilizzare lo script inserito nel CD allegato alla rivista. Dopo aver creato la tabella cliccateci sopra, con il tasto destro del Mouse, e quindi selezionate la voce di menu nuova Query SQL. Sulla finestra che compare inse-

Conoscenze richieste
Conoscenze su Moduli di Classe, oggetto MsHFlexGrid, oggetti ADO, SQL.

Software
Piattaforma Windows 2000 o superiore, Visual Basic 6 SP6, Microsoft dotNET Framework v2.0, SQL Server Express 2005.

Impegno

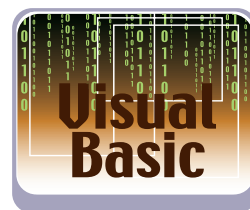
Tempo di realizzazione

Supporto	
Campo	**Tipo**
supportoID	int IDENTITY(1,1) NOT NULL
Descrizione	Nvarchar (250) NULL
Tipo	Nvarchar (50) NULL
Durata	Nvarchar (50) NULL
Critica	ntext NULL
Data	Datetime NULL
Trailer	Nvarchar (250) NULL
Masterizzato	Nvarchar (2) NULL
Copertinaimm	Image NULL
Posizione	Nvarchar (250) NULL
Lingua	Nvarchar (50) NULL
Una Tabella Complementare	
Campo	**Tipo**
supportoID	int NOT NULL
AttoreID	int NOT NULL
DesAttore	Nvarchar (250) NOT NULL

Tabella 1: Le tabelle Supporto ed Attore

rite il seguente codice e cliccate Execute.

```
Create PROCEDURE
[dbo].[SP_Select_Max_Supporto]
  (@supportoID [int]  output)
AS
  BEGIN
   SELECT @supportoID =  Max(supportoID) from
Supporto
              END
```

La *SP_Select_Max_Supporto*. La *SP* non fa altro che eseguire una query che *seleziona il massimo valore contenuto nel campo supportoId della tabella Supporto*.

2 Per Invocare la *SP_Select_Max_Supporto*, in un progetto *Visual Basic*, bisogna referenziare la *libreria ADODB - Microsoft Activex Data Object 2.6 Library* -, collegarsi al *database* con un *DSN* (*Data Source Name*. Il *DSN* va nominato *DSNdvd*. Le procedure che permettono di collegarsi e scollegarsi al database sono le seguenti:

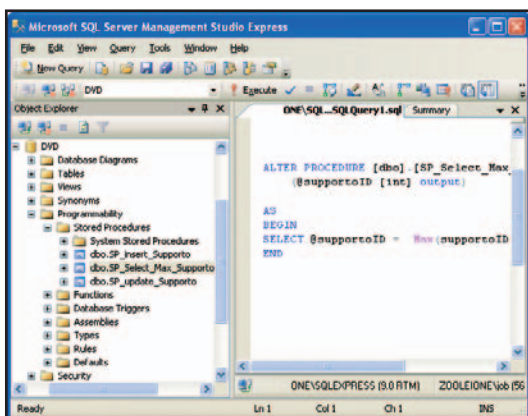



Fig. 1: Le Stored Procedure del DataBase DVD

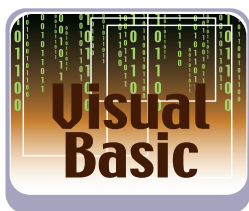
```
 Dim connectionADO As ADODB.Connection
Public Function Collegati() As ADODB.Connection
 Set connectionADO = New ADODB.Connection
 With connectionADO
  .ConnectionTimeout = 30
  .CommandTimeout = 100
  .Open "DSNdvd", "sa", ""
 End With
 Set Collegati = connectionADO
End Function
Public Sub Scollegati()
 connectionADO.Close
 Set connectionADO = Nothing
End Sub
```

RELAZIONI TRA TABELLE

Le informazioni contenute nelle tabelle di un *database* sono ricollegate attraverso le *relazioni* che sono delle associazioni tra *campi comuni* delle tabelle. I campi comuni sono: la *chiave primaria* (*Primary Key*) e la *chiave esterna* (*Foreign Key* – un campo con lo stesso nome della *Primary Key*) di un'altra tabella. Quest'ultima assicura l'*integrità referenziale* tra i dati delle tabelle. Le *Relazioni* possono essere:
- **uno-a-uno (cioè ad ogni *record* in una tabella può corrispondere un solo *record* nell'altra);**
- **uno-a-molti (ad ogni elemento della prima tabella possono corrispondere *n* elementi dell'altra)**
- **molti-a-molti (un elemento di una tabella può avere molti elementi corrispondenti nell'altra tabella e viceversa).**

3 Infine bisogna inserire il seguente codice in un pulsante.

```
Private Sub Command1_Click()
 Dim SpCom As ADODB.Command
 Dim par1 As ADODB.Parameter
 Dim strSql As String
 On Error GoTo errore
 Set SpCom = New ADODB.Command
 strSql = "SP_Select_Max_Supporto"
 Set par1 = SpCom.CreateParameter
          ("MaxsupportoID", adInteger, adParamOutput, 4)
 SpCom.Parameters.Append par1
 SpCom.CommandText = strSql
 SpCom.CommandType = adCmdStoredProc
 SpCom.ActiveConnection = Collegati
 SpCom.Execute
 MsgBox "Il valore Massimo è: " + CStr(SpCom.
                         Parameters("MaxsupportoID").Value)
 Set SpCom = Nothing
 Scollegati
 Exit Sub
errore:
 MsgBox Err.Description
End Sub
```

La procedura precedente invoca la *SP_Select_Max_Supporto* e mostra il valore massimo, restituito dalla *SP*, in un *MsgBox*.

L'APPLICAZIONE GESTIONE MEDIATECA

Nel precedente appuntamento abbiamo presentato un prototipo di applicazione per gestire un archivio con le informazioni sul contenuto di *supporti multimediali* (DVD, CD ecc.). Tra le informazioni trattate abbiamo incluso anche l'immagine della copertina (*cover*) del supporto, che abbiamo salvato in un campo di tipo *image* della tabella *Supporto*. In questo appuntamento modifichiamo radicalmente al tabella Supporto ed introduciamo altre tabelle complementari, collegati con relazione uno-a-molti alla Supporto. Queste tabelle includono soltanto delle descrizioni (qualcuno potrebbe dire che basterebbe una solo tabella con tanti campi descrizione e un tipo!?). Per esempio se il contenuto di un supporto multimediale è un film, le tabelle complementari potrebbero contenere l'elenco degli attori, dei registi ecc. Mentre se si tratta di *Software* le tabelle complementari potrebbero contenere informazioni sugli *Autori* dei programmi, sul tipo di licenza, sulla versione ecc. Tra le nuove *features* dell'applicazione inseriamo la gestione di informazioni multimediali contenuti in dei file compatibili con *Media Player*. Per archiviare questi file *Multimediali* (che potrebbero essere anche interi *film*) ci serviremo di una *directory* di *Windows*, collocabile nell'*Hard Disk* o nel *supporto ottico*. Nella **Tabella 1** sono riportate la nuova tabella *Supporto* e una tabella complementare. Nei paragrafi successivi descriveremo tre pezzi di applicazione.

1) La *Form* che gestisce i dati principali dei Supporti Ottici.
2) Il Modulo di Classe - *ClsSupporto* – che fa da intermediario tra la *Form* e la tabella *Supporto*.
3) La *SP_Insert_Supporto* - una delle *Stored Procedure* associate alla tabella *Supporto*.

NOTE

SQL SERVER EXPRESS E TOOLS

Collegandovi al link http://go.microsoft.com/fwlink/?LinkId=31401 potete scaricare *SQL Server Express* e *SQL Server Management Studio Express*. Ricordiamo che *SQL Express* può essere installato se è presente almeno *Windows 2000* e la versione *2.0* di *dotNET Framework*. Con *SQL Express* è installata, anche, *SQL Native Client*: la nuova tecnologia di accesso ai dati. Ricordiamo, però che si può accedere ai database *SQL Express* anche con il *Driver SQL Server* classico. Per approfondire le conoscenze su *SQL Server 2005* vi consigliamo di scaricare *SQL Server Books Online*.

LA FORM SUPPORTO

La *Form* che gestisce i dati principali dei supporti ottici è nominata *FrmSupporto*. Essa interagisce con i dati attraverso gli elementi pubblici della classe *ClsSupporto*. La *FrmSupporto* è divisa in tre parti: *Ricerca; gestione elementi tabella Supporto* e *gestione dati tabelle Complementari* (*Attori*, *Genere* ecc.). Nella parte superiore, dedicata alla ricerca, è presente una *MSHFlexGrid* collegata ad un *RecordSet Gerarchico* realizzato con una maschera *DataEnvironment*. La parte centrale, dedicata agli elementi della tabella *Supporto*, contiene gli oggetti grafici, indirettamente, collegati ai campi della tabella *Supporto*. In particolare i campi: *SupportoID*, *Descrizione*, *Posizione*, *Durata* sono associati a degli oggetti *textbox*; *Tipo* e *Lingua* sono associati a dei *ComboBox*; *Masterizzato* è associato a due *Option Button* (Si e No). Il campo *Immagine copertina*, invece, è associato ad una *PictureBox* ed a

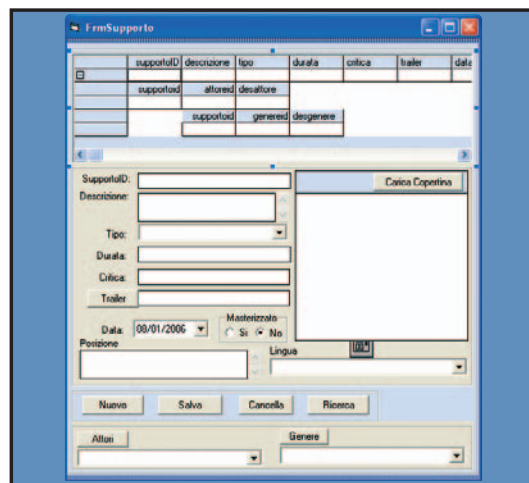



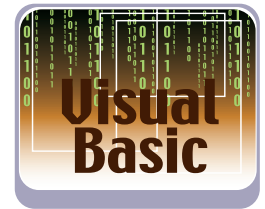
Fig. 2: **La FrmSupporto in fase di progettazione**

un *Command Button*. Quest'ultimo insieme ad un controllo *Common Dialog* permette di selezionare il file immagine da salvare nel *database*. Infine per gestire il file multimediale – il cui *Path* è salvato nel campo *Trailer* - sono previsti un *TextBox* e un *Command Button*. Quest'ultimo invoca un'altra *form* (nominata *frmDVD*) che riproduce il file con un oggetto *Media Player*. I dati di questi oggetti possono essere recuperati od inseriti nel *Database*, grazie ai pulsanti: *Nuovo*, *Salva*, *Cancella* e *Ricerca*. In realtà, come capiremo più avanti, il pulsante *Ricerca* è superfluo dato che gli elementi del *database* sono selezionati direttamente sulla *MSHFlexGrid*. La parte inferiore della Finestra, invece, permette di associare al *Supporto* corrente i valori delle *Tabelle Complementari*, queste funzionalità, però, al momento sono gestite solo in visualizzazione (se nelle tabelle *Complementari* sono presenti dei valori saranno mostrati nei *ComboBox* relativi). Del codice della *FrmSupporto* introduciamo soltanto quello strettamente legato al modulo di Classe *ClsSupporto* e alla *SP_Insert_Supporto*. Iniziamo a descrivere il codice partendo dalla parte dichiarativa che deve includere la definizione di un oggetto del tipo *ClsSupporto* e la costante che individua la *Directory* che contiene i file multimediali dei *Trailer*.

```
Dim BufSupporto As ClsSupporto
Const DirTrailer = "C:\Trailer\"
```

La *Form_Load* invece contiene il codice che carica i *ComboBox*: *Lingua* e *Tipo Supporto*

```
Private Sub Form_Load()
 CaricaCombo
End Sub
```

Quando si deve inserire un nuovo *Supporto* nell'archivio, sulla maschera bisogna innanzitutto cliccare il pulsante *Nuovo*, poi, specificare i valori per i vari elementi dell'interfaccia ed infine cliccare il pulsante *Salva*. Il codice per i pulsanti *Nuovo* e *Salva* è il seguente.

```
Private Sub Nuovo_Click()
 Set BufSupporto = New ClsSupporto
 txtsupportoID.Enabled = False
 SvuotaText
 'svuota gli elementi dell'interfaccia
 txtdescrizione.SetFocus
End Sub
Private Sub Salva_Click()
 'inserire controllo errori
 ValorizzaBuffer
 BufSupporto.salvaSP
 ValorizzaCampiForm
 'aggiorna gli elementi dell'interfaccia
 Set BufSupporto = Nothing
 SvuotaText
 Exit Sub
End Sub
```

La *ValorizzaBuffer*, invocata dalla *Salva_Click*, valorizza l'oggetto *BufSupporto* con i valori specificati negli elementi dell'interfaccia. Facciamo notare che nella *ValorizzaBuffer*, presentata sotto, non è presente l'attributo di classe associato all'immagine della copertina, questo perché la copertina è passata alla classe utilizzando il file immagine temporaneo *C:\temp.jpg* come vedremo nel codice della classe.

```
Private Sub ValorizzaBuffer()
 With BufSupporto
  .descrizione = txtdescrizione
  .tipo = ComboTipo.Text
  .Trailer = txttrailer
  .data = txtdata
  .critica = txtcritica
  .durata = txtdurata
  If Option1 Then
   .masterizzato = "SI"
  Else
   .masterizzato = "NO"
  End If
  .lingua = ComboLingua.Text
  .descrizionepos = txtposdescrizione
 End With
End Sub
```

La *Cancella Click()* e la *Ricerca Click()* sono le seguenti.

```
Private Sub Cancella_Click()
 On Error GoTo errore
 BufSupporto.Cancella
 Set BufSupporto = Nothing
 SvuotaText
Exit Sub
errore:
MsgBox Err.Description
End Sub
Private Sub Ricerca_Click()
 'inserire controlli errori
```

```
 Set BufSupporto = New ClsSupporto
 SvuotaText
 BufSupporto.Ricerca
 If BufSupporto.DaDB Then
  ValorizzaCampiForm
 Else
  MsgBox "Record non trovato", _
  vbOKOnly + vbCritical, "Attenzione"
 End If
End Sub
```

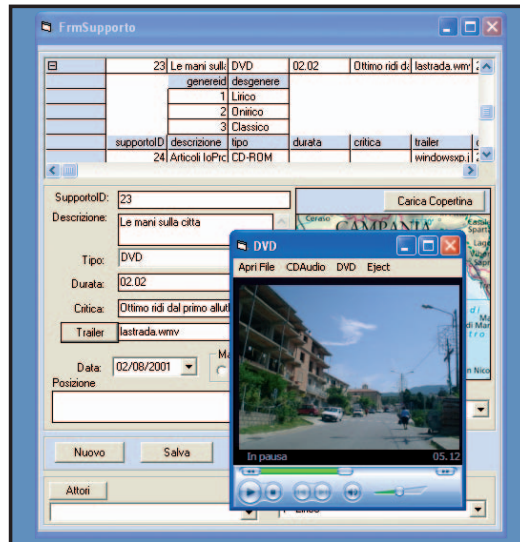



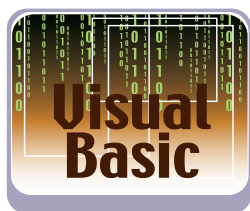
Fig. 3: La FrmSupporto in fase di esecuzione

L'attributo *DaDB*, della classe *ClsSupporto*, utilizzato nella *Ricerca_Click*, permette di stabilire se i dati presenti sulla maschera sono nuovi o letti dal database. Infine il codice che esegue il file multimediale sulla *FrmTrailer* è il seguente.

```
Private Sub Trailer_Click()
 FrmTrailer.avviatrailer DirTrailer + txttrailer
End Sub
```

La *FrmTrailer* e il resto del codice lo trovate nel CD allegato alla rivista.

TOOL DATA VIEW E T-SQL DEBUGGER

Nell'*IDE* di *Visual Basic* per gestire e controllare una Stored Procedure abbiamo a disposizione due strumenti: il *Data View* e il *T-SQL Debugger*. Il *Data View*, permette di connettersi al database e manipolare i suoi elementi. Il *T-SQL Debugger* permette di eseguire il *Debug* delle *Stored Procedure*. Il *Data View* si attiva cliccando l'icona (che rappresenta un cilindro di colore giallo) presente sulla *toolbar dell'ID*. Sulla finestra *Data View*, bisogna selezionare l'icona *Data Link* per creare un collegamento ad un database. Il collegamento può essere fatto attraverso l'*ODBC* oppure direttamente attraverso *OLEDB*. Per attivare il *T-SQL Debugger*, invece, basta selezionare, sulla finestra del *Data View*, una *Stored Procedure* e cliccare, con il tasto destro del *Mouse*, la voce di menu *Debug*. Il *T-Sql Debugger* permette: di vedere lo *stack* delle chiamate verso *SQL Server*, i valori delle variabili locali e globali e di controllare il flusso di programma nella *Stored Procedure*.

LA TECNICA DI RICERCA USATA NELLA FRMSUPPORTO

In questo esempio spieghiamo come visualizzare, con un oggetto *MSHFlexGrid*, i dati della gerarchia *Supporto, Attori* e *Generi* e come selezionare l'identificatore di un riga della griglia. Questa tecnica è usata per eseguire la ricerca sulla *FrmSupporto*.

1 Create un nuovo progetto *Visual Basic* ed in esso inserite una finestra di progettazione *Data Environment* ed un riferimento al componente *MSChart Control 6.0 (MSCHRT20.OCX)*.

Fig. 4: ***La Finestra Data Environment in cui è creata la gerarchia***

2 Attraverso un *DSN* (per esempio *DSNdvd*) collegate la finestra *Data Environment* al database *DVD* (che contiene le tabelle *Supporto*, *Genere* ed *Attore*). Con il *Data Environment* create una struttura gerarchica con tre *Command: Supporto, Attore* e *Genere*. Dove Supporto è il Parent delle altre due tabelle.

3 Sulla *Form1* inserite un oggetto *MsHFlexGrid*. Questo va collegato al *Data Environment* con il seguente codice.

```
Private Sub Form_Load()
 With MSHFlexGrid1
  Set .DataSource = DataEnvironment1
  .BandExpandable(0) = True
  .BandExpandable(1) = True
  .BandExpandable(2) = True
  .ColHeader(1) = flexColHeaderOn
  .ColHeader(2) = flexColHeaderOn
  .BandDisplay = flexBandDisplayVertical
 End With
End Sub
```

Le istruzioni precedenti, dopo aver associato la gerarchia del *DataEnvironment1* alla *MSHFlexGrid1* impostano le proprietà che permettono di espandere i dati in verticale.

4 Ora introduciamo le istruzioni che permettono di ricavare il valore del codice associato al *record* della tabella *Parent* della gerarchia (nel nostro caso il valore di *SupportoID* della tabella *Supporto*). A tal fine inseriamo le istruzioni nella *MSHFlexGrid1_Click()*.

```
Private Sub MSHFlexGrid1_Click()
 Dim com As Integer
 On Error Resume Next
 com = MSHFlexGrid1.RowSel
 While MSHFlexGrid1.TextMatrix(com, 1) <>
                                      "supportoID"
  com = com - 1
 Wend
 MsgBox MSHFlexGrid1.TextMatrix(com + 1, 1)
End Sub
```

LA TECNICA DI RICERCA USATA NELLA FRMSUPPORTO

Come accennato la *FrmSupporto* è correlata al modulo di Classe *ClsSupporto* che a sua volta è definita in base ai campi delle tabella *Supporto*. In particolare gli attributi della classe sono i seguenti:

```
Private mvarsupportoID As Long
Private mvardescrizione As String
Private mvartipo As String
Private mvardurata As String
Private mvarcritica As String
Private mvartrailer As String
Private mvardata As Date
Private mvarMasterizzato As String
Private mvardescrizionepos As String
Private mvarlingua As String
```

Nella classe definiamo anche l'attributo *DaDB* e la variabile globale *connectionADO*. *DaDB* serve a stabilire quale *SP* (la *Insert* o la *UpData*) eseguire nel metodo *SalvaSP*.

```
Private mvarDaDB As Boolean
Private connectionADO As ADODB.Connection
```

Di seguito riportiamo la definizione delle proprietà *Let* e *Get* dell'attributo *DaDB*.

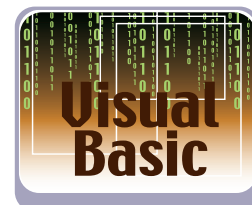
```
Public Property Let DaDB(ByVal vData As Boolean)
   mvarDaDB = vData
End Property
Public Property Get DaDB() As Boolean
   DaDB = mvarDaDB
End Property
```

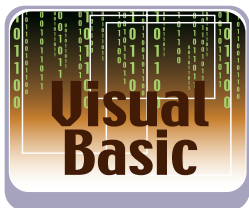
AMMINISTRAZIONE ODBC E DSN

L'Amministratore Origini dati ODBC permette di creare i DSN (Data Source Name). Ad esso si accede attraverso il menu strumenti di amministrazione del sistema Operativo o dal Pannello di controllo. Le connessioni alle fonti dati (per esempio Access o SQL Server) possono essere fatte se si è in possesso dei driver necessari (si controlli la scheda DSN utente dell'amministratore). Nel caso di SQL Express deve esserci il Driver SQL Native Client. Per creare un DSN basta seguire gli Step del Wizard Aggiungi DSN Utente questo permette di scegliere il Driver, il tipo e il nome del database. Per i nostri esempi abbiamo creato il DSN Utente nominato DSNdvd connesso al file C:\data\ dvd.mdf.

***Fig. 5:** La Finestra Creazione Guidata Modulo di Classe*

Ricordiamo che i moduli di classi possono essere creati utilizzando l'aggiunta *Creazione guidata Classe*, mostrata in figura 5. La *ClsSupporto* presenta diversi metodi, di seguito riportiamo il *SalvaSP* e il *CaricaCombo*. Il *SalvaSP*, in base al parametro *DaDB*, permette di fare un inserimento o un aggiornamento sul database. *CaricaCombo*, invece, permette di caricare un *ComboBox* con i dati di una tabella complementare.

```
Public Sub salvaSP()
Dim SpCom As ADODB.Command
Dim par1 As ADODB.Parameter
Dim strSql As String
Set SpCom = New ADODB.Command
If DaDB = True Then
 strSql = "SP_update_Supporto"
 Set par1 = SpCom.CreateParameter("supportoID",
                 adInteger, adParamInput, 4, supportoID)
 SpCom.Parameters.Append par1
Else
 strSql = "SP_insert_Supporto"
 Set par1 = SpCom.CreateParameter("supportoID",
           adInteger, adParamInputOutput, 4, supportoID)
 SpCom.Parameters.Append par1
End If
SpCom.CommandText = strSql
SpCom.CommandType = adCmdStoredProc
Set par1 = SpCom.CreateParameter("descrizione",
               adVarChar, adParamInput, 250, descrizione)
 SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("tipo", adVarChar,
                              adParamInput, 50, tipo)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("durata", adVarChar,
                            adParamInput, 50, durata)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("critica", adChar,
                        adParamInput, 1024, critica)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("trailer", adVarChar,
                          adParamInput, 255, trailer)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("data",
                adDBTimeStamp, adParamInput, 14, data)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("masterizzato",
                adChar, adParamInput, 2, masterizzato)
SpCom.Parameters.Append par1
Dim stado As New ADODB.Stream
 `oggetto introdotto nel precedente articolo
 stado.Type = adTypeBinary
 stado.Open
 stado.LoadFromFile "c:\temp.jpg"
 Set par1 = SpCom.CreateParameter("copertinaimm",
            adLongVarBinary, adParamInput, stado.Size,
                                          stado.Read)
 SpCom.Parameters.Append par1
 stado.Close
 Kill ("c:\temp.jpg")
Set par1 = SpCom.CreateParameter("posizione",
         adVarChar, adParamInput, 250, Me.descrizionepos)
SpCom.Parameters.Append par1
Set par1 = SpCom.CreateParameter("lingua", adVarChar,
                              adParamInput, 50, lingua)
SpCom.Parameters.Append par1
SpCom.ActiveConnection = Collegati
SpCom.Execute
If DaDB = False Then
 `nuovo record
   Me.supportoID = SpCom.Parameters("supportoID").Value
   DaDB = True
End If
Set SpCom = Nothing
Scollegati
Exit Sub
errore:
 MsgBox Err.Description
End Sub
Public Sub CaricaCombo(combo As ComboBox)
 Dim RecordsQuery As New ADODB.Recordset
 Dim cSQL As String
 combo.Clear
 Select Case combo.Name
  Case "ComboAttori"
   cSQL = "select * from attore" _
```

IL MODELLO THREE-TIER

Il *Three-Tier* è un modello di sviluppo per applicazioni *Client/Server*. Esso stabilisce che gli elementi di un'applicazione devono essere distribuiti su tre livelli o strati: *User Tier, Business Tier, Data Tier*. In ogni strato gli elementi devono svolgere una determinata funzione e scambiare, se necessario, dati e risultati con gli altri elementi dello stesso strato e con quelli dello strato inferiore o superiore. Ogni livello può essere costituito da uno o più componenti residenti sullo stesso computer o distribuiti su computer diversi. L'*User Tier* è lo strato responsabile dell'interfaccia utente. Il *Business Tier*: contiene gli elementi che implementano il cuore dell'applicazione, cioè le regole con le quali devono essere elaborate le informazioni presenti sul database prima di essere mostrati attraverso lo strato di presentazione. Il Data Tier é lo strato responsabile dell'accesso e del mantenimento dei dati persistenti dell'Applicazione, esso si estende fino al *DBMS*.

```
    & " where supportoID = " & CStr(supportoID)
  Case "ComboGenere"
     cSQL = "select * from genere" _
    & " where supportoID = " & CStr(supportoID)
  End Select
  RecordsQuery.Open cSQL, Collegati, adOpenKeyset, _
  adLockOptimistic """" adOpenForwardOnly
  While Not RecordsQuery.EOF
   combo.AddItem CStr(RecordsQuery.Fields(1)) _
   & " - " + RecordsQuery.Fields(2)
   RecordsQuery.MoveNext
  Wend
  If combo.ListCount <> 0 Then
   combo.ListIndex = 0
  End If
End Sub
```

Sul CD allegato alla rivista, trovate la classe completa e il metodo *Salva* implementato senza *SP*.

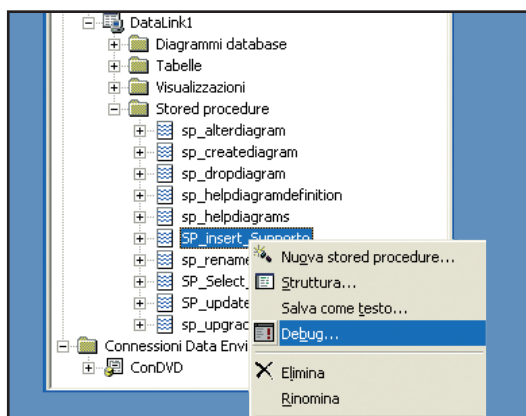



Fig. 6: **Come avviare il Debug di una Stored Procedure dal Visualizzatore Dati**

UNA SP DELLA TABELLA SUPPORTO

In genere una *SP* è associata ad una singola azione che è possibile fare su una tabella, per esempio si definisce la *SP_Insert* per inserire un record, la *SP_Update* per modificarlo e la *SP_Cancella* per cancellarlo. Nel caso della tabella *Supporto* definiamo: *SP_insert_Supporto*, *SP_Update_Supporto* e *SP_Delete_Supporto*. Le *Stored Procedure* le possiamo implementare manualmente come illustrato in precedenza oppure utilizzando il *Wizard - Create Stored Procedure* - presente in *SQLServer*, ma non in *SQL Express*. In quest'ultimo infatti sono presenti soltanto dei *Template* di *SP*. Di seguito presentiamo la *SP_insert_Supporto*. Dato che la *Tabella Supporto* ha 11 campi, la *SP* presenta 11 parametri di cui uno (*SupportoID*) di solo *output*.

```
CREATE PROCEDURE [SP_insert_Supporto]
 (@supportoID [int]  output,
  @descrizione [nvarchar](250),
  @tipo [nvarchar](50) ,
  @durata [nvarchar](50) ,
  @critica [ntext] ,
  @trailer [nvarchar](255),
  @data [datetime] ,
  @masterizzato [nvarchar](2) ,
  @copertinaimm [image] ,
  @posizione [nvarchar](250),
  @lingua [nvarchar](50))
   as
  Begin
    INSERT INTO [supporto]
     ( [descrizione]
      ,[tipo]
      ,[durata]
      ,[critica]
      ,[trailer]
      ,[data]
      ,[masterizzato]
      ,[copertinaimm]
      ,[posizione]
      ,[lingua]
      )
    VALUES
      (@descrizione,
       @tipo,
       @durata,
       @critica,
       @trailer,
       @data,
       @masterizzato,
       @copertinaimm,
       @posizione,
       @lingua
      )
    SELECT @supportoID = @@IDENTITY
  end
```

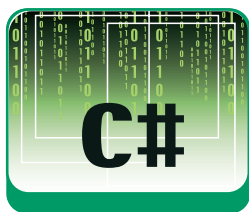
Nella *SP* prima è lanciata una *query* di *Insert* e poi è ricavato il valore di *supportoID* (che ricordiamo è un campo che crea una sequenza numerica per identificare i nuovi record). L'identificatore del record è restituito da *@@IDENTITY*, questo è impostato nel parametro di *Output @supportoID*. Notate che la parola chiave *Select* è usata sia per interrogare il database che per assegnare dei valori ai parametri.

CONCLUSIONE

Gli argomenti introdotti in questo articolo – modello *Three-Tier, Moduli di Classe, Stored Procedure* ecc. - meritano un accurato approfondimento dato che costituiscono le fondamenta delle applicazioni *Client/Server* … buon lavoro.

Massimo Autiero

CREARE CONTROLLI PERSONALIZZATI CON .NET

IL .NET FRAMEWORK PERMETTE LA CREAZIONE DI CONTROLLI NUOVI PER LA PERSONALIZZAZIONE DELLE APPLICAZIONI WINDOWS, CON IL SUPPORTO DI AVANZATI MECCANISMI DI LICENSING PER LA LORO DISTRIBUZIONE

Lo sviluppo di custom controls, o controlli personalizzati, è un tema chiave del framework .NET. In generale essi rendono l'architettura di un'applicazione più riutilizzabile e portabile, in quanto lo stesso controllo, progettato come si deve, esponendo la giusta dose di proprietà per la sua personalizzazione, potrà essere riutilizzato senza fatica in diverse applicazioni. Creare controlli custom in .NET è notevolmente più semplice dell'implementazione che un tempo, sempre più lontano, veniva fatta tramite controlli ActiveX in C++ o in Visual Basic. Controlli che spesso portavano al classico problema conosciuto come *DLL Hell*, in quanto ogni versione doveva essere registrata nel sistema operativo, per non parlare degli altri problemi che si incontravano In .NET creare un *custom controls* non differisce molto dal creare una classe qualunque, basta ereditare dalla classe madre giusta.

TIPI DI CUSTOM CONTROLS

Ci sono diverse possibilità per creare dei controlli personalizzati da utilizzare nelle nostre applicazioni, il modo più frequente è quello di derivare un controllo dalla classe *UserControl* ed utilizzare una sorta di composizione di altri controlli, standard oppure personalizzati anch'essi. Un'altra modalità e quella di derivare dalla classe *Control*, ed utilizzare GDI+ per disegnare l'interfaccia grafica. Questi controlli, detti anche *owner-drawn*, sono quelli più customizzabili, ma naturalmente richiedono più sforzi di sviluppo. La classe *Control* infatti è più alta nella gerarchia di ereditarietà rispetto alla classe *UserControl:*

```
System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Windows.Forms.Control
        System.Windows.Forms.ScrollableControl
          System.Windows.Forms.ContainerControl
            System.Windows.Forms.UserControl
            System.Windows.Forms.Form
```

Notate come *UserControl* sia allo stesso livello della classe *Form*, quindi come vedremo ne condivide molte funzionalità e comportamenti, ed il fatto che derivi da *ContainerControl* permette di contenere altri controlli proprio come una Form.
Se si vuole invece estendere un controllo esistente con nuove funzionalità, si può derivare direttamente dalla sua classe e personalizzarlo secondo le diverse esigenze. Naturalmente non esiste una linea guida da seguire in maniera precisa, ognuno si comporterà come meglio ritiene al momento di sviluppare un nuovo controllo.

IL SUPPORTO DI VISUAL STUDIO.NET 2005

Visual Studio .NET 2005, ma anche le versioni precedenti, forniscono un ottimo supporto nella realizzazione e nell'utilizzo di *custom controls*.
Il primo passo è creare un nuovo progetto, e scegliere come tipologia *Windows Control Library* oppu-

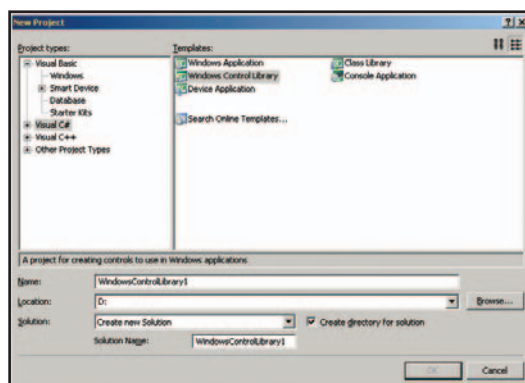



Fig.1: Creare un progetto di controlli

re *Class Library* (vedi **Figura 1**).
Non fa molta differenza, in quanto entrambi permetteranno di creare un assembly .NET, da referenziare nelle nostre applicazioni. Il primo comunque

Fig.2: Aggiungere uno user control al progetto

creerà anche un file *UserControl1.cs*, scheletro di classe da utilizzare per creare un controllo derivante da *UserControl*. In ogni caso, per aggiungere un nuovo *User-Control* al progetto, basta cliccare con il tasto destro, nella finestra *Solution Explorer*, sul nome del progetto, e selezionare la voce *Add/User Control* (**Figura2**). Il designer di Visual Studio permette a questo punto di progettare il controllo aggiungendo elementi standard sulla sua superficie, altri *usercontrols*, e di impostarne le classiche proprietà, ad esempio dimensione, colore di sfondo, e così via, in maniera perfettamente analoga a ciò che siamo abituati a fare con una Form. Visual Studio 2005 ha aggiunto la possibilità di avviare un progetto di tipo *Windows Control Library*. Infatti all'avvio del debug viene creato un contenitore dei controlli contenuti nella libreria attraverso il quale interagire con esso e testarlo (**Figura 3**).

Fig.3: Il test container di VS2005 per gli user control

Per testare il nostro usercontrol nella edizione 2003 avremo invece bisogno di una applicazione *Windows Forms* che lo utilizzi. Quindi aggiungiamo alla soluzione un nuovo progetto *WindowsApplication*, chiamandolo ad esempio *TestApplication*. Perché l'applicazione possa utilizzare il controllo, dobbiamo aggiungere fra i riferimenti l'assembly dell'altro progetto. In questo caso dopo aver cliccato su *References*, bisogna selezionare la scheda *Projects* e selezionare il progetto contenente lo *UserControl*. Già questo è sufficiente per utilizzare nel vostro codice il controllo personalizzato. Se volete anche la comodità di trascinarlo dalla barra degli strumenti data uno sguardo al box apposito che spiega come personalizzare sia la toolbox che l'icona che contraddistingue il controllo.

Fig.4: Aggiungere il riferimento alla libreria dei controlli

L'IPADDRESSCONTROL

Una volta creato il progetto *Windows Control Library*, è possibile iniziare la personalizzazione del nostro controllo. Supponiamo di voler realizzare un controllo che permetta l'inserimento e la convalida di un indirizzo IP, nel formato classico *aaa.bbb.ccc.ddd*, cioè 4 gruppi di numeri da 0 a 255. Basta inserire sulla superficie del controllo 4 textbox, una label, ed aggiungere un errorProvider per indicare l'inserimento di valori non consentiti. Per consentire solo l'inserimento di valori numerici, basterà gestire l'evento *KeyPress*, con un metodo che associeremo a tutte quante le *TextBox:*

```
private void textBoxes_KeyPress(object sender,
                                KeyPressEventArgs e)
{
    if (char.IsDigit(e.KeyChar) || e.KeyChar ==
                                (char)Keys.Back)
    {   e.Handled = false; }
     else e.Handled = true;
}
```

invece per verificare che i valori inseriti nelle caselle siano compresi fra 0 e 255, intanto limiteremo la lunghezza massima delle stringhe inserite a 3 caratteri, impostando la proprietà *MaxLength* di *TextBox*. Inoltre, gestendo l'evento *TextChanged*, si potrà verificare che il valore di ogni *TextBox* sia nel range consentito, ed in caso negativo impostare *l'errorProvider.*

```
private void textBoxes_TextChanged(object sender,
                                EventArgs e)
{
    TextBox txt = sender as TextBox;
    try
    {
        if (txt.Text == "")
            errorProvider.SetError(txt, "");
        else
        {
            int i = Convert.ToInt32(txt.Text);
            if (i < 0 || i > 255)
            { errorProvider.SetError(txt, "Inserire un
```

```
                          valore fra 0 e 255"); }
            else errorProvider.SetError(txt, ""); } }
    catch
    { errorProvider.SetError(txt, "Inserire un
                          valore fra 0 e 255"); }
}
```

Ogni user control può naturalmente definire le proprie proprietà pubbliche, che potranno essere usate anche nel designer di VisualStudio. In questo caso, ad esempio, creiamo una proprietà *IPAddress* che restituisca l'IP impostato nel controllo, sotto forma di stringa, e viceversa per effettuare un parsing di una stringa nel formato classico di un indirizzo IP:

```
public string IPAddress
{  get
    { return String.Format("{0}.{1}.{2}.{3}",
                    textBox1.Text, textBox2.Text,
                    textBox3.Text, textBox4.Text); }
    set
    {
        try
        {
            string[] values = value.Split('.');
            textBox1.Text = values[0];
            textBox2.Text = values[1];
            textBox3.Text = values[2];
            textBox4.Text = values[3]; }
        catch
        {  IPAddress = "..."; }
    }
}
```

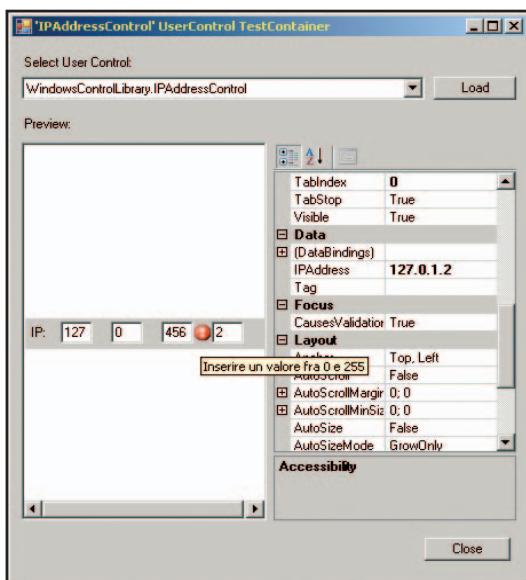



Fig.5: *L'IPAddressControl nel test container*

La **Figura 5** mostra il controllo nel *TestContainer*, con l'error provider attivo sul terzo blocco di cifre, e la proprietà *IPAddress* visibile nella *PropertyGrid* di destra. Se il controllo custom possiede delle proprietà pubbliche, che però non volete rendere visibili nella griglia delle proprietà, a design time, basta applicare l'attributo *Browsable* alla proprietà con parametro *false*, ad esempio:

```
[Browsable(false)]
public string IPAddress
...
```

Un altro attributo utile è *Category*, che permette di visualizzare una proprietà in una determinata categoria della griglia. Ad esempio se si vuol visualizzare la proprietà precedente nella categoria *Data*, basterà scrivere:

```
[Category("Data")]
public string IPAddress
```

CONTROLLI OWNER DRAWN

Una seconda possibilità di sviluppo di controlli custom, è quella che prevede l'intero disegno del controllo con le classi ed i metodi *GDI+*. In questa maniera si ha la possibilità di una personalizzazione molto più spinta e quindi di poter progettare e realizzare controlli completamente nuovi. Supponiamo ad esempio di voler utilizzare nella nostra applicazione dei pulsanti rotondi. Le strade sono due, o compriamo un controllo già fatto, o ce lo costruiamo da soli. Scegliamo la seconda. Intanto bisogna creare una nuova classe, chiamandola ad esempio *RoundButton* e derivandola da *Control*. Per il nostro pulsante vogliamo inoltre la possibilità di impostare i colori del bordo, quello di sfondo, cambiando quest'ultimo quando esso viene cliccato e quando il puntatore del mouse ci si muove sopra. Dunque alla classe aggiungeremo i campi necessari a mantenere tali possibili colori e stati:

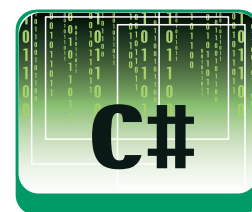
```
public class RoundButton : Control
{
        private Color borderColor;
        private Color fillColor;
        private Color hoverColor;
        private Color clickedColor;
        private bool clicked;
        private bool hover;
}
```

Per i campi *Color* è utile aggiungere le relative proprietà public, in maniera da utilizzarle anche a *Design Time*, quando si aggiungerà il pulsante su una Form. Ad esempio la proprietà *FillColor* sarà la seguente:

```
[RefreshProperties(RefreshProperties.Repaint),
```

• PROFESSIONAL C#
Robinson et al.
(Wrox Press)

• PROGRAMMING WINDOWS WITH C#
Petzold
(Microsoft Press)

• USER INTERFACES IN C#: WINDOWS FORMS AND CUSTOM CONTROLS
MacDonald
(Apress)

```
                    Category("Appearance"),
                Description("Colore di riempimento")]
public Color FillColor
 {
   get
        { return fillColor; }
        set
        {
           fillColor = value;
           Invalidate();
        }
 }
```

Il metodo che effettua il disegno di un controllo è il metodo *OnPaint*, ed è dunque necessario effettuarne l'override, distinguendo le diverse modalità di disego del pulsante, a seconda dello stato in cui esso si trova:

```
protected override void OnPaint(PaintEventArgs e)
{
        Graphics graphics = e.Graphics;
        int penWidth = 4;
        Pen pen = new Pen(borderColor, 4);
        SolidBrush brush;
        if (clicked)
           brush = new SolidBrush(clickedColor);
        else
        {
           if (hover)
              brush = new SolidBrush(hoverColor);
           else
           { brush = new SolidBrush(fillColor); }
        }
        int x = clicked ? 2 : 0;
        graphics.FillEllipse(brush, x, x, Width-x,
                                            Height-x);
        SolidBrush textBrush = new SolidBrush(
                                           ForeColor);
        graphics.DrawEllipse(pen, (int)penWidth / 2+x,
           (int)penWidth / 2+x, Width - penWidth-x,
                                  Height - penWidth-x);
        graphics.DrawString(Text,Font, textBrush,
              penWidth, Height / 2 - Font.Height);
     }
```

Ciò che modificherà lo stato del *RoundButton* sarà naturalmente l'interazione dell'utente, per mezzo del mouse. Quindi basterà gestire in maniera classica gli eventi del mouse e variare i valori dei campi della classe. Ad esempio quando il puntatore del mouse entra sulla superficie del pulsante, basterà impostare a true il campo hover, e forzare un ridisegno invalidando la superficie del controllo. Viceversa, quando il mouse abbandona l'area del pulsante si riporterà a *false* tale campo:

```
private void RoundButton_MouseEnter(object sender,
```

```
                                          EventArgs e)
{
        hover = true;
        Invalidate(); }
private void RoundButton_MouseLeave(object sender,
                                          EventArgs e)
{
        hover = false;
        Invalidate();
}
```

La **Figura 6** mostra una Form con due *RoundButton*, uno dei quali è evidenziato al passaggio del mouse.

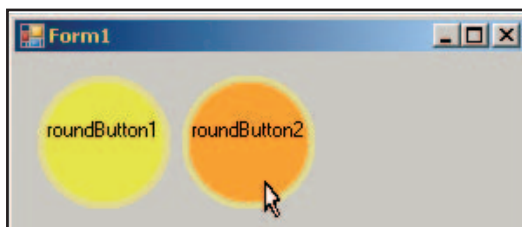



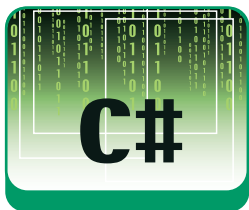
***Fig. 6:* Pulsanti tondi**

LICENSING DEI CONTROLLI

Una volta che abbiamo creato il nostro bel controllo, e siamo pronti a distribuirlo, magari vorremmo anche avere la possibilità di difenderci dall'utilizzo non autorizzato, cioè implementare qualche meccanismo di Licensing, che sia valido non solo a runtime, ma anche a design time. Il .NET framework possiede un modello di licensing integrato, utilizzabile sia per controlli windows che per web controls, ed inoltre totalmente compatibile con gli ActiveX. La validazione della licenza viene svolta in questo modello da una classe che deriva dalla classe astratta *LicenseProvider*. La classe *LicFileLicenseProvider* è una semplice implementazione che utilizza un file di licenza fornito assieme al controllo. Naturalmente se avete in mente di distribuire commercial-

AGGIUNGERE IL CONTROLLO ALLA TOOLBAR

Per aggiungere un custom control alla barra degli strumenti di Visual Studio, basta cliccare con il tasto destro sulla toolbox, e selezionare la voce *Choose Items*, a questo punto, fate clic sul pulsante *Browse* e ricercate l'assembly che contiene il controllo. Una volta selezionatolo cliccate su *OK* ed esso apparirà sulla barra. Se desiderate personalizzare l'icona di default (la ruota dentata), non dovete far altro che aggiungere un file bitmap al progetto, nelle sue proprietà impostare *embedded resource* come *build action* e quindi aggiungere l'attributo *ToolboxBitmap* alla classe del controllo, specificando il percorso dell'immagine nell'assembly, ad esempio:

```
[System.Drawing.ToolboxBitmap(typeof(
                         IPAddressControl),
              "TuoNamespace.Immagine.bmp")]
```

mente i vostri controlli, questo meccanismo non è sufficiente, ma è utile per mostrare il funzionamento del licensing. Il primo passo è utilizzare l'attributo *LicenseProvider,* in questo caso con parametro *LicFileLicenseProvider*, applicandolo alla classe che implementa il controllo, ed aggiungere un campo *License.*

```
[LicenseProvider(typeof(LicFileLicenseProvider))]
public partial class RoundButton : Control
{
   private License license = null;
 …
}
```

Nel costruttore della classe è ora necessario invocare il metodo statico *Validate* della classe *LicenseManager,* che, se tutto è ok, restituira un oggetto *License* con cui inizializzare il campo license.

```
public RoundButton()
{
  …
  license = LicenseManager.Validate(
                                typeof(RoundButton), this);
}
```

Il metodo Validate, se non trova un file di licenza valido, genera una eccezione *LicenseException,* impedendo dunque la creazione del custom control, sia a runtime che a design-time, dato che anche il designer internamente cerca di creare un'istanza del controllo. Infine, nel finalizzatore della classe, o comunque prima dell'esecuzione del finalizzatore, è necessario invocare il metodo *Dispose* del campo license:

```
if (license != null)
{
          license.Dispose();
          license = null;
}
```

Non resta che creare il file di licenza. Ma prima provate ad utilizzare il vostro controllo in un progetto Windows Forms, e verificate che viene generata l'eccezione con un messaggio del tipo:

```
Exception occurred creating type
                    'WindowsControlLibrary.RoundButton,
                  WindowsControlLibrary, Version=1.0.1.0,
                    Culture=neutral, PublicKeyToken=null'
        System.ComponentModel.LicenseException: An
    instance of type 'WindowsControlLibrary.RoundButton'
          was being created, and a valid license could
                           not be granted for the type
        'WindowsControlLibrary.RoundButton'. Please,
         contact the manufacturer of the component
                                  for more information.
```

In parole povere manca un file di licenza valido. Con l'implementazione standard *LicFileLicenseProvider* basta creare un file con nome uguale al nome completo di namespace della classe che implementa il controllo ed estensione *.LIC,* nel nostro caso sarà un file *WindowsControlLibrary.RoundButton.LIC,* ed inserire il testo seguente:

```
WindowsControlLibrary.RoundButton is a licensed
                                             component.
```

Non dimenticate il punto finale. Tale file deve essere presente nella directory di output, quella cioè dove visual studio compila l'eseguibile. In Visual Studio 2005, basta impostare la proprietà *Copy to output* del file al valore *Copy always*. Se provate adesso a ricompilare la soluzione, vedrete che va tutto a posto. Come già detto un file di licenza così semplice, e con un testo predefinito, non è il massimo della sicurezza. Quindi bisogna adottare un meccanismo di verifica del file più forte, ad esempio ereditando dalla classe *LicFileLicenseProvider* una nuova classe, ed effettuando l'override del metodo *IsKeyValid:*

```
class MyFileLicenseProvider:LicFileLicenseProvider
   {
        protected override bool IsKeyValid(string key,
                                              Type type)
        {
            if (key.IndexOf("123456789")>=0)
                return true;
            return false;
         }
   }
```

Il metodo *IsKeyValid* prende in ingresso il parametro *key,* che sarà inizializzato al contenuto del file *LIC*. In questo caso il metodo restituisce true se il file contiene la stringa *"123456789".* A questo punto potete sbizzarrirvi a creare modalità di verifica più complesse del file di licenza, oppure derivando una classe direttamente dalla classe astratta *LicenseProvider* e quindi senza essere legati per forza ad un file.

CONCLUSIONI

Abbiamo visto come procedere nell'implementazione di un controllo personalizzato per applicazioni Windows Forms, derivandolo dalla classe *UserControl,* e quindi come composizione di controlli standard, o disegnando by scratch la sua interfaccia, arrivando infine ai meccanismi di licensing dei componenti e controlli .NET. Siete pronti ad iniziare la vostra avventura nel mondo dei *custom controls*?

Antonio Pelleriti

CONTATTA L'AUTORE

Potete rivolgere domande di chiarimenti o ulteriori richieste all'autore all'indirizzo antonio.pelleriti@ioprogrammo.it **o ancora sul forum di ioProgrammo** http://forum.ioprogrammo.net **o sul sito** www.dotnetarchitects.it

MSMQ: OVERVIEW ARCHITETTURALE

MSMQ È IL SERVIZIO WINDOWS CHE FACILITA LA COMUNICAZIONE ASINCRONA FRA APPLICAZIONI PER TUTTI GLI SCENARI IN CUI LE PARTI POSSONO ESSERE IRRANGIUNGIBILI OFFLINE O LAVORARE A VELOCITÀ DIVERSE

Quante volte abbiamo cercato qualcuno al telefono e lo abbiamo trovato al primo colpo? Quante volte abbiamo ritentato la telefonata senza successo perdendo minuti preziosi ? La soluzione "telefonica" al problema citato è arrivata un bel po' di tempo fa con l'invenzione delle segreterie telefoniche: *"Marco, sono Roberto, ti volevo comunicare che domani la riunione è spostata alle 17"*. La filosofia delle comunicazioni asincrone è più o meno questa.
Registriamo su un supporto il messaggio che l'interlocutore "off-line" o impegnato potrà ascoltare non appena disponibile. I due interlocutori non parlano direttamente tra loro, ma tramite messaggi memorizzati in uno store (nastro nel caso delle segreterie telefoniche). La posta elettronica è un esempio informatico concreto di comunicazione asincrona in quanto i due interlocutori non sono direttamente connessi, non devono conoscere la posizione fisica l'uno dell'altro, ma si scambiano informazioni tramite un servizio che si preoccupa di instradare il messaggio fino a raggiungere la casella postale del destinatario. Per definizione l'email è asincrona: non possiamo, ne dovremmo, aspettarci una risposta immediata; possiamo solo inserire un flag di priorità in modo che il destinario possa discriminare i vari messaggi da processare. La posta elettronica è un ottimo sistema di scambio di informazioni fra "umani", ma dimostra subito i suoi limiti quando viene impiegata per altri scopi, quali lo scambio di dati fra applicazioni.

REQUISITI

Conoscenze richieste
nessuna

Software
.NET Framework 1.x o 2.0 per testare il codice presentato

Impegno

Tempo di realizzazione

PERCHÈ MSMQ ?

MSMQ si basa sugli stessi principi, ma è stato progettato nello specifico per comunicazioni *application-to-application* e si propone come servizio per lo scambio di messaggi fra applicazioni, fornendo maggior velocità, semplicità e soprattutto garanzie rispetto alla posta elettronica.

QUANDO?

Pensiamo ad un sito web pubblico che consente l'acquisto online; se la comunicazione con l'istituto di credito è interrotta o troppo lenta come ci comportiamo?
Le strade sono due:

1) Ci scusiamo con l'utente chiedendogli se per favore può riprovare più tardi. Più tardi quanto? 1 minuto? 10 minuti? 1 giorno? Fra quanto la linea sarà di nuovo "in palla" come dovrebbe. E se la transazione è lenta? Facciamo aspettare l'utente 5 minuti sulla pagina?

2) Accodiamo la richiesta da qualche parte, informiamo l'utente che il suo acquisto è in fase di "processing" e che riceverà una comunicazione sull'esito della transazione.

Se fossi io a progettare il sito di commercio elettronico, proporrei la seconda soluzione per vari motivi, non solo commerciali:

- Ci garantiamo acquisti anche in caso di caduta di linea (motivo commerciale)
- Non rallentiamo la fase di acquisto per lentezza del partner commerciale o della linea
- Non blocchiamo un thread (preziosissimo) per qualche minuto se la linea è lenta: si veda il box laterale:

Un altro esempio per allargare lo scenario: pensiamo ad un agente commerciale che

raccoglie ordini direttamente dal cliente tramite il suo palmare. Ogni ordine deve essere trasmesso alla sede; anche in questo caso abbiamo due strade:

1) Trasmettiamo l'ordine immediatamente con il rischio che, se la linea GPRS/UMTS non è disponibile, l'agente debba prendere per mano il cliente, uscire dall'azienda, cercare insieme una zona coperta per inviare l'ordine, tornare nello stabile, prendere un altro ordine e uscire di nuovo mano nella mano con il cliente per premere il pulsante *"Invia"*

2) Ogni ordine viene memorizzato localmente e inviato quando l'agente ritrova la connettività.

Fig. 1: ***I servizi da installare per provare le tecniche descritte***

A prescindere dalle tecnologie disponibili per gestire questi semplici scenari, in entrambi i casi la soluzione **2** è sicuramente preferibile. Arriviamo subito al punto: dove memorizziamo queste richieste da inviare "non appena possibile" (potrebbe voler dire che la sera l'agente invia tutti gli ordini della giornate se non è indispensabile processarli il prima possibile)? Spesso la soluzione che ci inventiamo è costruire una tabella in un database locale dove memorizzare le informazioni (non l'ordine) sul cliente, gli articoli ordinati, il tipo di pagamento scelto etc) che verrano spedite a qualche entità server side (Web Service per esempio) al momento in cui ritroviamo la connettività. Dobbiamo gestire noi stessi la frequenza di invio, l'eventuale priorità con cui spedire i record (forse ha più senso processare prima gli ordini di clienti importanti oppure gli acquisti di maggior valore) e soprattutto le politiche transazionali con cui inviare il messaggio. Quest'ultimo è il punto più importante: quando cancello il record dalla tabella locale? Come faccio a essere sicuro che sia arrivato a destinazione? Quando levo il record dalla tabella di destinazione? Quando sono sicuro che l'applicazione remota lo abbia processato. E se l'operazione non va a buon fine? Ad esempio non si può prendere l'ordine perché il cliente è insolvente: come avverto l'applicazione iniziale sull'esito dell'operazione?

LE CARATTERISTICHE DI MSMQ

Facciamo entrare in gioco un prodotto nato per gestire queste problematiche: *Microsoft Message Queue*. È disponibile in versione 1.0 dal 10 marzo 1997, si è evoluto in versione 2 e 3 su Windows 2000/2003/XP e dal 1999 è disponibile anche per piattaforma Windows CE 3.0. L'idea è molto semplice: il *Sender* scrive un messaggio all'interno di una coda gestita dal servizio e il *Receiver* andrà a leggere il messaggio dalla medesima coda.
MSMQ garantisce il recapito, il *routing* (instradamento), la sicurezza durante il trasporto e fornisce interfacce di programmazione per accedere in lettura/scrittura alle code.
Il prodotto si colloca in una realtà distribuita in cui le comunicazioni fra i vari componenti software che costituiscono l'applicazione girano su macchine diverse, macchine che possono essere connesse da linee non sempre attive (*Dial-up, VPN, GPRS,* ecc.), oppure quando è necessario far comunicare applicazioni che girano su sistemi operativi differenti e/o non si vuole che problemi di rete possano fermare o rallentare l'esecuzione. Le comunicazioni asincrone risolvono anche l'annoso problema delle performance.
Già, perché il tempo necessario per scrivere un messaggio in una coda può essere considerato quasi uguale a zero, mentre attendere

THREAD PREZIOSISSIMI IN ASP.NET?

Già, i *thread* sono risorse preziosissime per tutti gli ambienti server-side. ASP.NET, per default, dispone di 25 thread (per ogni processore su macchine Dual Core o Multiprocessore) per processare tutte le richieste Http relative alle applicazioni ospitate nello stesso processo (in IIS 5 tutte le applicazioni ASP.NET sono ospitate nello stesso processo).
Premesso che tale numero si può alzare da configurazione, liberare il prima possibile un thread significa avere più risorse per processare le richieste. Non è mai una buona idea aumentare il numero di thread se il server non riesce a processare le richieste in quanto il costo di thread switch è elevato. A tal proposito si veda un'introduzione a *IHttpAsyncHandler* e *ASP.NET 2.0 Async* si veda: http://blogs.devleap.com/articolidevleap/archive/2006/01/12/6505.aspx

una comunicazione sincrona con un'applicazione più lenta, su una connessione di rete lenta, significa rallentare anche l'applicazione chiamante. Pensiamo anche, ad esempio, alle problematiche di replica dei dati da un database installato su quadriprocessore Xeon con 8 Gb di RAM verso un database di bakcup situato su una macchina Pentium IV monoprocessore con 1 Gb di RAM: la nostra super macchina è costretta a "rallentare" per adeguarsi alla velocità del Pentium IV; ulteriore rallentamento si avrebbe se le due macchine fossero connesse con un collegamento non troppo veloce oltre al fatto che comunque le due macchine debbano entrambe rimanere online per tutta la durata della replica.In una comunicazione sincrona le richieste sono processate nell'ordine in cui vengono inviate dal chiamante, mentre con un sistema di messaggistica è possibile assegnare una priorità ad ogni messaggio.

Fig. 2: **L'interfaccia di amministrazione**

Un crash nella comunicazione sincrona causa un rollback, cioè un annullamento di tutte le operazioni effettuate (e molto spesso un messaggio di errore per l'utente), mentre MSMQ può eseguire un numero di retry impostabile dall'applicazione. È possibile definire il tempo massimo il cui il messaggio può raggiungere la coda di destinzione e un tempo massimo in cui il messaggio deve essere letto. In caso di timeout è il servizio MSMQ che si incarica di inserire un messaggio amministrativo di mancato recapito o mancata lettura che, sempre tramite le API, è possibile recuperare e interpretare.

Nel caso in cui un certo numero di client effettuano richieste sincrone di aggiornamento dati occorre definire una politica di lock sulle risorse rallentando o terminando l'operazione di aggiornamento nel caso in cui la risorsa sia bloccata. Una coda non ha nessun problema di concorrenza in quanto i messaggi vengono inseriti e accodati dalle varie applicazioni *Sender*. L'applicazione *Receiver* processerà i messaggi uno ad uno e quindi effettuerà degli aggiornamenti "sequenziali" sulle risorse. Quest'ultimo punto è discutibile in quanto potrebbero esistere più applicazioni *Receiver* per ragioni di performance o *Fault-tolerance*, o più semplicemente il *Receiver* potrebbe girare multithread facendo riemergere il problema dei lock sulla risorsa finale. Anche in questo caso però abbiamo spostato il problema dall'applicazione che l'utente usa (applicazione web, windows o mobile) all'applicazione receiver, liberando così il front-end dalle problematiche descritte.
L'applicazione sender può scrivere in una coda locale o in una coda remota senza preoccuparsi della connettivtà verso la coda. MSMQ tiene il messaggio in locale per poi spedirlo nella coda remota appena viene ritrovata la connettività: non occorre quindi definire una coda locale sul palmare dell'agente commerciale visto nel caso precedente; l'applicazione scriverà sempre nella coda aziendale e sarà MSMQ a incaricarsi della spedizione non appena il palmare verrà connesso alla struttura aziendale.
L'architettura del prodotto consente configurazioni molto complesse integrate con domini, sedi remote, bridge verso prodotti di terze parti, integrazione con *Active Directory*, ma proviamo a partire dal caso più semplice: un sito web che accoda delle richieste che vengono poi "scodate" da un'applicazione che si incarica di effettuare il vero lavoro.
Torneremo poi sui dettagli architetturali e altri scenari in un prossimo articolo.

PRIME PROVE

Per iniziare è sufficiente installare su macchina singola (non complichiamoci la vita con macchine diverse per adesso, ribadisco però che la location delle code è indipendente) il MSMQ presente fra i componenti di Windows. Una volta installato il prodotto da *"Computer Management" -> Services and Applications* appare la voce *Message Queuing*. Sotto *Private Queue* si può creare amministrativamente una nuova coda assegnando

un nome qualunque. Le code private si raggiungono dalle applicazioni (*Sender* o *Receiver*) tramite il proprio path corrispondente a *"nomemacchina\private$\nomecoda"*. Si possono usare vari formati e protocolli, e si possono creare code anche da codice (avendo il permesso di farlo) ma per adesso concentriamoci sul caso più semplice. Per default una coda ad accesso a *Everyone* quindi per fare qualche prova non ci dovrebbero essere problemi di security.

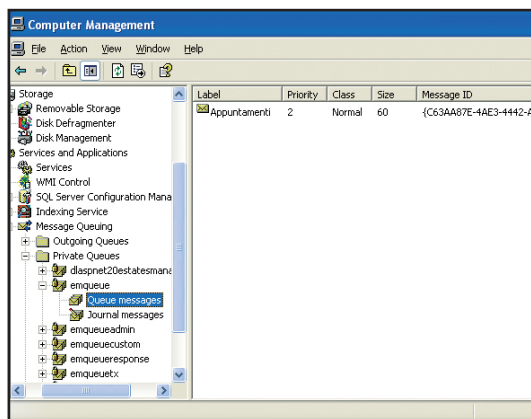



Fig. 3: ***Visualizza i messaggi in una coda***

Un'applicazione che vuole inviare un messaggio in coda dovrà referenziare la libreria *.NET System.Messaging* (sia in .NET 1.x che 2.0), creare un oggetto *MessageQueue* che rappresenta la coda e un oggetto *Message* che rappresenta il messaggio da spedire nella coda. Ecco un estratto di codice C# di un'applicazione Windows Form che invia un messaggio nella coda *"gondor\private$\testmsmq"* dove in questo caso *"gondor"* è il nome della mia macchina.

```
…
using System.Messaging;
…
try
{
    using(MessageQueue mq = new MessageQueue(
                        "gondor\private$\testmsmq"))
    {

        Message messaggio = new Message();
        messaggio.Body = "Ciao dal Sender";
        mq.Send(messaggio);
    }
    MessageBox.Show("Inviato!");
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

L'esempio, molto semplice, utilizza il pattern *C# using* per eseguire la *Dispose* dell'oggetto che rappresenta la coda e il tipo *MessageQueueException* per intercettare eventuali problemi durante l'invio del messaggio.
Se il messaggio viene recapitato in coda senza errori, sempre da *Computer Management*, eseguendo un refresh su *Queue Message* nella coda si potrà visualizzare e analizzare il messaggio. Come si può notare al messaggio viene assegnato un ID, viene riportato il sender, le informazioni sull'arrivo e il body del messaggio stesso. In questo caso il contenuto del messaggio è la stringa indicata appunto nel body dall'applicazione sender. Il messaggio, per default, viene serializzato in XML all'interno della coda: non c'è nessun bisogno di convertire la stringa in XML quanto il Formatter di default (cioè colui che si incarica di serializzare il messaggio) è XmlMessageFormatter. Parleremo di serializzazione XML e di classi custom in un prossimo articolo.
L'applicazione che deve ricevere i messaggi e processarne il contenuto utilizzerà sempre la classe MessageQueue per indicare la coda da cui prelevare il messaggio e il metodo Receive per estrarre il messaggio dalla coda.
Ecco il codice:

```
…
using System.Messaging;
…
try
{
    using(MessageQueue mq = new MessageQueue(
                        "gondor\private$\testmsmq"))
    {
        mq.Formatter = new XmlMessageFormatter(
                    new Type[] { typeof(String) });
        // Ricezione del messaggio
        Message messageReceived = mq.Receive();
        String contenuto = (String)
                        messageReceived.Body;
    }
}
catch (MessageQueueException ex)
{
    MessageBox.Show("Errore invio");
}
```

Per ricevere correttamente un messaggio è necessario indicare il tipo di contenuto (tipo inteso nel senso .NET: la classe) che il *Message Formatter* si incaricherà appunto di formattare. Per prima cosa, quindi, nel nostro caso, indichiamo all'oggetto *MessageQueue* che il suo formatter è un *XmlMessageFormatter* che dovrà trattare il tipo *String* per dese-

Potete rivolgere domande di chiarimenti o ulteriori richieste all'autore agli indirizzi:

www.DevLeap.com
http://blogs.devleap.com/rob
http://thinkmobile.it

L'AUTORE

Roberto Brunetti è un libero professionista del gruppo DevLeap (www.devleap.com) sul cui sito si trovano articoli e blog tecnici sulle tecnologie legate allo sviluppo software in .NET. È specializzato in ASP.NET, Sviluppo mobile, Architetture distribuite e Visual Studio Team System. È l'autore del libro *ASP.NET Full Contact* edito da Mondadori Informatica e numerose pubblicazioni su riviste del settore. Ha partecipato come speaker a numerose conferenze del settore.

rializzare il contenuto del *Body*. Il metodo *Receive* restituisce un oggetto di tipo *Message* corrispondente al messaggio estratto dalla coda. È sufficiente poi ricovertire il corpo del messaggio nel suo tipo (classe) originale. Il metodo *Receive* elimina il messaggio dalla coda... scoda in poche parole. In questo esempio, il contenuto del messaggio è una semplice stringa che "non dice molto" all'applicazione che riceve il messaggio: è importante capire che il contenuto del messaggio è a cura dello sviluppatore (o meglio del progettista della soluzione basata su MSMQ) e quindi non esistono messaggi preconfezionati. Il body deve essere significativo per l'applicazione Sender e per l'applicazione *Receiver*.

Fig. 4: ***Il body del messaggio memorizzato in XML all'interno di una coda***

Ad esempio un body reale di una mia vecchia applicazione è *"APPT:36; DAY:2001/01/09; TIME:12-00;DURATION:2"* che indica un appuntamento per il giorno 9/1/2001 alle ore 12.00 per due ore con ID 36. In questo caso la stringa viene composta e spacchettata dalle due applicazioni. Nella versione successiva dell'applicativo tale contenuto viene invece espresso in XML tramite la serializzazione automatica della classe *Appuntamento* definita fra le entità del progetto. Adesso che abbiamo capito la filosofia di base di Microsoft Message Queue e visto il codice per spedire e ricevere un messaggio, vediamo alcune opzioni di delivery che consentono di dare una priorità ai messaggi, di tracciare la spedizione e la ricezione. In molti scenari ha senso suddividere le tipologie di messaggio da spedire/ricevere in base a diversi criteri; potremmo, ad esempio, dover inviare messaggi relativi a ordini da processare e messaggi relativi a spedizioni da effettuare: in questo caso è opportuno creare due code diverse in cui recapitare i messaggi in modo da rendere totalmente indipendenti i messaggi, la loro spedizione, e soprattutto la loro ricezione: l'applicazione o le applicazioni che devono analizzare il contenuto del messaggio apriranno la coda ordini se devono processare ordini o la coda spedizioni per analizzare le spedizioni da effettuare. Anche all'interno della stessa coda (*Ordini* ad esempio per tornare allo scenario di inizio articolo) potremmo però aver bisogno di differenziare fra ordini urgenti e ordini meno urgenti. La priorità è una delle proprietà impostabili sul singolo messaggio da spedire: è l'applicazione o le applicazioni sender che impostano tale proprietà; i messaggi nella coda di arrivo verranno "scodati" (termine tecnico per indicare il *Receive* del messaggio) in base a tale priorità. La proprietà è denominata *Priority*. È importante capire che la priorità non è assoluta: voglio dire che non è assolutamente certo che un messaggio a priorità bassa venga per forza di cose letto dopo un messaggio con priorità alta. Il motivo è semplice: se un'applicazione spedisce un messaggio a priorità bassa e subito dopo un messaggio a priorità alta e l'applicazione receiver è libera (e esiste connettività verso la coda) riceverà subito il messaggio a priorità bassa che verrà processato prima del messaggio successivo. Questo comportamento non rappresenta un problema e non deve rappresentarlo: se sembra un problema significa che non abbiamo inquadrato bene la problematica di lavoro asincrona. Prendiamo anche il caso in cui due Sender mandano nello stesso istante un messaggio a priorità alta e uno a priorità alta: se ad esempio il messaggio a priorità alta è più pesante (in termini di Kb) è probabile che raggiunga la coda di destinazione dopo il messaggio a priorità bassa. Ripeto: questo comportamento non è un problema e non deve esserlo.

CONCLUSIONI

Per questo primo articolo direi di fermarci: abbiamo cercato di chiarire con due esempi il ruolo di un sistema di messaggistica applicativa per poi vedere la forma più semplice di invio e ricezione di un messaggio. Abbiamo ancora molto da capire su MSMQ: metodi di delivery, garanzia di spedizione, transazionalità, journaling, administration queue, invio di classi custom all'interno dei messaggi.

Roberto Brunetti

UN ASSO PER IL MULTIPIATTAFORMA

CON LE LIBRERIE ACE È POSSIBILE REALIZZARE IN MANIERA ESTREMAMENTE SEMPLICE APPLICAZIONI CHE COMUNICANO VIA RETE. WINDOWS O LINUX NON FA DIFFERENZA! VEDREMO COME FARE TUTTO SENZA DIFFICOLTÀ

Spesso ci si trova di fronte alla necessità di dovere realizzare un'applicazione di rete in C/C++. Ci si pone di conseguenza la seguente domanda: *"Sarà per Windows o per Linux?"*. Al contrario di altri linguaggi di programmazione, come Java ad esempio, la scelta del sistema operativo è di cruciale importanza per definire quella che sarà la struttura del codice, oltre che il codice stesso. La comunicazione di dati via rete, infatti, si basa moltissimo sulle funzioni messe a disposizione dal sistema operativo sottostante. Un programma, per quanto semplice, che debba avere le stesse funzionalità su entrambe le piattaforme più diffuse, avrà per forza di cose codice diversissimo. Questa diversità tuttavia cozza un po' con quelli che sono i concetti di "riusabilità" e "generalità" che caratterizzano i software di buona fattura. Il concetto di "connessione TCP/IP", infatti, è unico e non ha senso distinguere tra *"connessione TCP/IP con Linux"* e *"connessione TCP /IP con Windows"*. La distinzione è introdotta a livello pratico dalle intrinseche differenze dei due sistemi. Sarebbe molto bello, in linea teorica, realizzare una libreria generica che consenta, ad esempio, di utilizzare una ipotetica classe *ConnessioneTCPIP* e lasciare ad essa il compito gravoso di gestire le chiamate di sistema giuste. Ancora più bello sarebbe trovare una libreria di questo tipo già fatta, di ottima qualità e semplice utilizzo! Nel prosieguo dell'articolo si parlerà di questa opportunità.

REQUISITI

Conoscenze richieste
Basi di C++

Software
ACE, un compilatore C++

Impegno

Tempo di realizzazione

COS'È ACE

ACE sta per *ADAPTIVE Communication Environment* ed è un potentissimo insieme di librerie scritte in C++ per aiutare lo sviluppatore a scrivere facilmente applicazioni che siano:

- **portabili:** le interfacce utilizzate sono le medesime per ciascun sistema operativo. L'unica cosa da fare per un corretto porting è utilizzare la corretta libreria ACE sottostante;
- **dalle alte prestazioni:** sono inclusi meccanismi di ottimizzazione dell'utilizzo della memoria. Ad esempio alcune classi permettono una gestione "interna" dei comandi *"new"* e *"delete"* in modo che l'allocazione sia fatta una sola volta e in quantità sufficiente , e non di volta in volta, con tutti i rallentamenti dovuti all'interazione col sistema operativo;
- **multithreading:** è semplicissimo avviare un nuovo thread. Esistono molti modi tra i quali quello di istanziare un oggetto di una classe derivata da *ACE_Thread* e definire opportunamente la *funzione svc()*;
- **per il networking:** grazie all'infrastruttura fornita è davvero istantaneo instaurare connessioni di rete sia di tipo TCP che UDP. Un particolare occhio di riguardo è riservato alle prestazioni, che non sono assolutamente intaccate dall'utilizzo di ACE

Insomma si tratta di uno strumento di sviluppo che qualsiasi programmatore dovrebbe imparare ad utilizzare in maniera costante, anziché reinventare ogni volta la ruota. ACE insomma non ha prezzo, e questo non è solo un apprezzamento riguardo le sue straordinarie doti tecniche, ma anche una constatazione del fatto che è completamente gratuito e libero da royalties di utilizzo!

L'APPLICAZIONE

Ci concentreremo in questo articolo sullo sviluppo di una piccola applicazione client-server che dimostri le potenzialità di ACE in ambito di networking. Per mantenere il tutto il più semplice possibile creeremo un semplice echo-ser-

ver, ovvero un programma che svolge le seguenti azioni:

- si mette in ascolto su una porta per l'arrivo di un messaggio
- quando il messaggio arriva, lo rimanda indietro identico (come l'eco...).

Per concentrarci meglio sulle funzionalità di base offerte da ACE e sul suo meccanismo di comunicazione, tralasceremo intenzionalmente tutte le problematiche relative alla realizzazione di un server "buono" dal punto di vista ingegneristico. In particolare il nostro server:

- **non gestirà connessioni multiple:** potrà essere connesso ad un solo client per volta;
- **non effettuerà controlli sul tipo di dato ricevuto:** potrebbe quindi ricevere stringhe potenzialmente dannose.

Al nostro server affiancheremo un'applicazione di tipo client, che non farà altro che mandare un messaggio al server e riceverne l'eco (cioè il messaggio così come è stato inviato). Si può benissimo capire come si tratti di esempi-giocattolo, che sono tuttavia validi dal punto di vista didattico. Vediamo dunque l'implementazione dei due programmi.

IL SERVER

Il codice del server viene riportato di seguito, commentato in tutte le sue parti. Innanzitutto è necessario includere gli header giusti, contenenti le classi che utilizzeremo di seguito:

```
#include "ace/INET_Addr.h"
#include "ace/SOCK_Stream.h"
#include "ace/SOCK_Acceptor.h"
#include <iostream>
```

Il primo file contiene la dichiarazione di *ACE_INET_Addr,* la classe utilizzata per le informazioni riguardanti un indirizzo IP, come ad esempio la porta, l'indirizzo vero e proprio ecc. *SOCK_Stream.h* contiene la dichiarazione di *ACE_SOCK_Stream,* l'oggetto che rappresenta un flusso di informazioni sul quale effettuare le operazioni di lettura/scrittura dei dati che vengono ricevuti/inviati utilizzando la rete. *SOCK_Acceptor.h* serve per potere utilizzare *ACE_SOCK_Acceptor,* ovvero la classe che permette di istanziare oggetti che si "mettono in ascolto" sulla rete, in attesa di potere accettare delle richieste. L'utilizzo di questa classe è peculiare di un programma di tipo server: tale programma ferma la sua elaborazione e attende delle richiesta da parte di altri. Al contrario un programma di tipo client, come vedremo, inoltra attivamente delle richieste e si aspetta che il server le soddisfi nel più breve tempo possibile. Seguono delle dichiarazioni di costanti che utilizzeremo in seguito:

```
#define MAXHOSTNAME 256
#define BUFFERSIZE 1024
#define PORT_NUMBER 12345
```

A questo punto inizia il programma vero e proprio con la definizione della funzione *main():*

```
int main (int argc, char* argv[])
{
  ACE_INET_Addr port_to_listen (PORT_NUMBER);
  ACE_SOCK_Acceptor acceptor;
```

INSTALLARE E CONFIGURARE ACE

1. I sorgenti Per la sua natura multipiattaforma ACE viene distribuito attraverso un pacchetto contenente i sorgenti del software. I sorgenti vanno compilati scegliendo l'opportuna configurazione per il proprio sistema operativo.

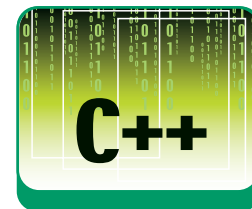
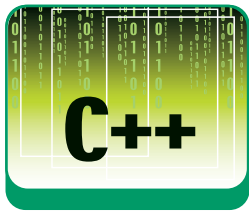
2. Compilare sotto Windows Sono supportati moltissimi OS e compilatori. Ad esempio utilizzando Visual C++ 6.0 sotto Win32, scegliamo di compilare ACE come una DLL. Per fare questo apriamo il file ACE_wrappers\ace\ace_dll.dsw

3. Scegliere la configurazione Nella cartella ACE_wrappers\ace effettuiamo una copia del file config-win32.h e la rinominiamo in config.h. Questo header contiene le opzioni giuste per generare la DLL.

4. Build. Torniamo all'interno del progetto aperto col Visual C++ e premiamo su "Build All". La compilazione richiede qualche minuto. Una procedura analoga è prevista anche per i sistemi Unix-like (ad es. Linux) e si basa sull'esecuzione di alcuni script Perl inclusi.

Vengono istanziati due oggetti. Il primo, *port_to_listen,* è un indirizzo di rete del tipo *ACE_INET_Addr* visto in precedenza. Viene creato specificando unicamente il numero di porta, per cui si riferirà alla macchina locale *(localhost).* Il secondo oggetto è di tipo *ACE_SOCK_Acceptor* e sarà quello che materialmente si occuperà di attendere una connessione in ingresso e di instaurarla quando sarà il caso. L'attesa di una connessione viene fatta occupando una porta libera del sistema operativo:

```
  if (acceptor.open (port_to_listen, 1) == -1)
  {
    std::cerr << "Non posso utilizzare la porta "
      << PORT_NUMBER << "! Esco..."
      << std::endl;
    return 1;
  }
```

Con ACE questa cosa si fa utilizzando la funzione

open() dell'oggetto *ACE_SOCK_Acceptor.* La porta desiderata va specificata come primo argomento, tramite l'oggetto *ACE_INET_Addr* di prima. Qualora il risultato di questa funzione fosse *-1* ci troveremmo in una situazione di errore. Tipicamente ciò avviene quando la porta è già utilizzata da un altro programma. Ad esempio lanciando due istanze del nostro server, la seconda darà inevitabilmente un errore in questo punto. A questo punto comincia il lavoro vero e proprio del server, cioè l'attesa di una nuova connessione e la successiva gestione della stessa:

```
int cont = 1;
std::cerr << "[" << cont << "] In attesa di una
                    connessione..." << std::endl;
  while (1)
{
    ACE_SOCK_Stream peer;
    ACE_INET_Addr peer_addr;
    ACE_Time_Value timeout (10, 0);
    if (acceptor.accept (peer, &peer_addr, &timeout,
                                          0) == -1)
    {
        std::cerr << "[" << ++cont << "] In attesa
                di una connessione..." << std::endl;
    }
```

All'interno di un ciclo infinito *(while(1) {...})* viene istanziato l'oggetto peer di tipo *ACE_SOCK_Stream*. Su questo oggetto verranno effettuate le operazioni di invio e ricezione dati. Viene creato inoltre un altro oggetto di tipo *ACE_INET_Addr,* che conterrà l'indirizzo del client remoto e la porta sul quale la connessione viene gestita. L'oggetto timeout è di tipo *ACE_Time_Value* e serve a far sì che la successiva chiamata

```
acceptor.accept()
```

abbia una "scadenza" dopo un certo periodo di tempo, in questo caso 10 secondi. Questo è necessario poiché la *accept()* è di tipo bloccante, cioè ferma l'esecuzione dell'intero programma fino a quando non viene istaurata una connessione. Questo comportamento di solito non è voluto in un server, ecco perché dopo la scadenza del timeout viene stampato un messaggio con un numero progressivo, ad indicare che il programma è "vivo". Successivamente viene eseguito un nuovo ciclo di attesa di una connessione. Quando la connessione ha successo viene gestita dal seguente codice:

```
    else
    {
        ACE_TCHAR peer_name[MAXHOSTNAME];
        peer_addr.addr_to_string (peer_name,
                                   MAXHOSTNAME);
        std::cout << "Connessione avvenuta con "
                      << peer_name << std::endl;
        char buffer[BUFFERSIZE];
        ssize_t bytes_received;
        while ((bytes_received = peer.recv (buffer,
                            sizeof(buffer))) != -1)
        {
            std::cout << "[" << cont++ << "]
                    RICEVO: " << buffer << std::endl;
            peer.send_n (buffer, bytes_received);
        }
        peer.close();
    }
  }
  return 0;
}
```

Le prime tre righe del ramo *"else"* non fanno altro che stampare a schermo l'indirizzo remoto col quale si effettua la connessione nonché la porta sulla quale le connessione è gestita (tale porta è normalmente diversa dalla porta di attesa della connessione). L'indirizzo viene ricavato semplicemente utilizzando la funzione *addr_to_string()* che restituisce una stringa nel formato:

```
INDIRIZZO:PORTA
```

Nel successivo ciclo *while* avvengono le operazioni di ricezione e invio dei dati. La ricezione è effettuata dall'istruzione *recv()* all'interno della condizione del *while* stesso. L'invio di ciò che è stato ricevuto avviene con la *send_n(),* nella quale specifichiamo il buffer da inviare e il numero di byte di cui esso è composto. Sia *recv()* che *send_n()* vengono richiamate, come già anticipato, sull'oggetto peer di tipo *ACE_SOCK_ Stream*. Possiamo provare subito il server effettuando una connessione telnet sulla porta 12345. Scriviamo ad esempio:

```
telnet localhost 12345
```

dopo avere lanciato l'eseguibile del server. A questo punto tutto ciò che scriviamo dovrebbe essere stampato 2 volte: 1 è il carattere scritto e 1 quello ricevuto dall'echo-server:

```
iiooPPrrooggrraammmmoo  ccoonn  ACCEE
```

IL CLIENT

Il codice del client è più semplice di quello del server. Il suo compito è quello di connettersi

al server sulla macchina locale alla porta conosciuta e inviare un messaggio con una cadenza regolare di 1 secondo tra un invio e l'altro. Il client stampa inoltre quello che riceve dal server.
Vediamo insieme il codice:

```
#include "ace/INET_Addr.h"
#include "ace/SOCK_Stream.h"
#include "ace/SOCK_Connector.h"
#include <iostream>
#define PORT_NUMBER 12345
#define MAXBUFSIZE 256
#define MESSAGE "Ciao! Sono il client..."
int main (int argc, char* argv[])
{
  ACE_INET_Addr server (PORT_NUMBER,
                                    ACE_LOCALHOST);
  ACE_SOCK_Connector connector;
  ACE_SOCK_Stream peer;
```

Inizialmente troviamo le consuete inclusioni analoghe al codice del server. Nella funzione *main()*. Vengono istanziati tre oggetti ACE. L'utilizzo dell'*ACE_SOCK_Stream* è identico a quello visto in precedenza: serve per le operazioni di lettura/scrittura. L'indirizzo contenuto nella variabile "server" indica l'IP della macchina su cui il server risiede, in questo caso la macchina locale. A differenza del codice del server, per stabilire la connessione di rete, non troviamo un oggetto di tipo *"acceptor"* bensì uno di tipo *"connector":*

```
  std::cout << "Cerco di connettermi al server sulla
                                              porta "
           << PORT_NUMBER << std::endl;
  if (connector.connect (peer, server) == -1)
  {
    std::cerr << "Impossibile connettersi al server!"
                                         << std::endl;
    return 1;
  }
  else
  {
    std::cout << "Connesso!" << std::endl;
  }
```

il connector cerca di stabilire la connessione, se non ci riesce (ad es. perché il server non è stato lanciato) restituisce subito un risultato di errore, senza ulteriori attese. In altre parole non c'è bisogno di prevedere l'utilizzo di un timeout, poiché si chiede attivamente una connessione, e non si deve restare in attesa di nulla. Per instaurare la connessione si utilizza la funzione *connect()*.

Come nel caso del server, anche per il client è previsto un ciclo *while* infinito che periodicamente invia un messaggio predefinito:

```
int bc, cont = 0;
  char buf[MAXBUFSIZE];
  char message[] = MESSAGE;
  while (1)
  {
     std::cout << "[" << ++cont << "] ";
     std::cout << "INVIO: " << message
                                    << std::endl;
     peer.send_n (message, sizeof(message));
     bc = peer.recv (buf, sizeof(buf));
     std::cout << "[" << cont << "] ";
     std::cout << "RICEVO: " << buf << std::endl;
     ACE_OS::sleep(1);
  }
  peer.close ();
  return (0);
}
```

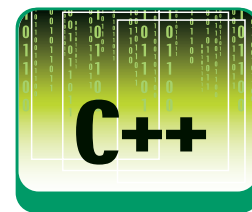
Le operazioni di invio/ricezione dati vengono effettuate, come visto in precedenza, attraverso le *recv()* e *send_n()*. Da notare come l'attesa del tempo prestabilito avvenga utilizzando la funzione ACE

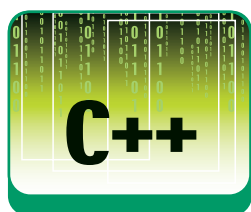
```
ACE_OS::sleep(1);
```

che generalizza le varie funzioni di attesa presenti nei diversi sistemi operativi. Questa è solo una delle tante utilità offerte dalla libreria ACE, che permettono di scrivere codice altamente portabile.
Come ultima cosa segnaliamo il fatto che, in genere, ogni flusso dati che si crea dovrebbe essere opportunamente chiuso, al termine del suo utilizzo mediante la *close()*.

CONCLUSIONI

In questo articolo abbiamo illustrato solo la punta dell'iceberg di quell'insieme di librerie software che è racchiuso sotto il nome ACE.
Sottolineiamo come queste librerie non solo offrano l'infrastruttura per scrivere un codice altamente portabile, utilizzabile sotto una grande varietà di sistemi operativi, ma consentano anche di fare le comuni operazioni di sistema in maniera molto più semplice del solito, senza alcuna perdita in termini di efficienza. Ci sentiamo insomma di raccomandare lo studio e l'utilizzo regolare di queste librerie, così come facemmo, su queste stesse pagine, nel caso delle STL.

Alfredo Marroccelli

SVILUPPARE IN C++ PER WINDOWS E LINUX

ALLA SCOPERTA DI WXWIDGETS, UNA POTENTE LIBRERIA CHE CI CONSENTE DI CREARE APPLICAZIONI MULTIPIATTAFORMA IN MODO SEMPLICE E VELOCE. IN QUESTO NUMERO LAVORIAMO CON LE CARATTERISTICHE AVANZATE

Qualche numero fa di ioProgrammo ci siamo occupati della realizzazione di interfacce grafiche tramite le librerie wxWidgets. Riassumiamo brevemente qualche concetto base. *WxWidgets* (anche conosciuta come *wxWindows*) è un framework open source per la realizzazione di interfacce utente cross-platform con aspetto nativo. Praticamente *wxWidgets* è un insieme di librerie che permette di realizzare applicazioni C++ in grado di essere compilate ed eseguite su un gran numero di piattaforme. Questo è possibile poiché sono disponibili tante versioni delle librerie per quante sono le piattaforme supportate. Le piattaforme grafiche attualmente supportate sono Windows, GTK+, Motif, e Mac. *wxWidgets* non copre solo gli aspetti relativi all'interfaccia grafica ma copre anche problematiche inerenti la gestione del file system, i servizi di rete, il multithreading ed il 3D tramite OpenGL. Gli strumenti necessari per utilizzare *wxWidgets* sono un compilatore C++ e la libreria stessa. Inizializzare una form con le *wxwigets* è quanto di più semplice al mondo: l'intera applicazione viene gestita da un oggetto di tipo *wxApp*. La classe wxApp è astratta e bisogna ridefinire due metodi OnInit e OnExit.

```
class myApp : public wxApp
{ public:
        bool OnInit();
        int OnExit();
};
```

quando andiamo ad implementare la classe myApp dobbiamo inserire la seguente riga :

```
IMPLEMENT_APP( myApp )
```

Poi andiamo a definire le funzioni.

```
bool myApp::OnInit()
{  wxFrame *myWin =
    new  wxFrame(NULL,-1,
      "Hello wxWorld");
   SetTopWindow(myWin);
   myWin->Show(TRUE);
   return TRUE; }
int myApp::OnExit(){ return 0; }
```

A myWin possono poi essere aggiunti dei controlli, ad esempio:

```
wxBoxSizer* box =
new wxBoxSizer(wxHORIZONTAL);
box->Add( new wxTextCtrl(myWin, ID_TEXTBOX ,
                  "Testo", wxPoint(0,0),wxSize(100,20) ),
             0,wxALIGN_CENTER_VERTICAL | wxALL, 5);
box->Add( new wxButton( myWin, ID_BOTTONE,
                  "Bottone",wxPoint(0,0),wxSize(80,20) ),
              0,wxALIGN_CENTER_VERTICAL | wxALL,5);
myWin->SetSizer(box);
```

e infine possiamo gestire gli eventi

```
class myFrame : public wxFrame
{  public:
   myFrame( wxWindow * parent,
                        int id, const wxString &title);
   void OnTextChange(wxCommandEvent &event );
   void OnPressButton(wxCommandEvent &event );
   private:
     DECLARE_EVENT_TABLE()
 };
```

Per ragioni di spazio non possiamo riassumere oltre quanto già detto nei numeri precedenti. In questo articolo ci occuperemo di alcun aspetti avanzati.

FILES E DIRECTORY

Immaginiamo per prima cosa di voler ottenere tutti i file e le directory contenute in percorso generico. L'espressione del percorso di ricerca dipende dalla piattaforma di esecuzione dell'applicazione finale. wxWidgets, comunque, riesce ad utilizzare il formato dei percorsi tipico di unix anche sotto altre piat-

taforme, senza dovere proteggere i percorsi con lunghe sequenze di escape.

```
#include <wx/dir.h>
wxDir dir( percorso );
if ( !dir.IsOpened() )
{ wxMessageBox("Il percorso non esiste");
   return; }
wxString filename;
bool continua = dir.GetFirst(&filename, "*.*" );
while ( continua )
{ //operazioni con filename
   continua = dir.GetNext(&filename);
}
```

La funzione GetFirst restituisce,attraverso il puntatore alla stringa filename, il nome del primo file che rispecchia la stringa di ricerca passatagli come secondo argomento. Se nessun file è stato trovato il valore restituito è false, altrimenti è true e la ricerca continua. I file successivi vengono restituiti dalla funzione GetNext. Per ottenere informazioni su i file trovati con il metodo appena illustrato possiamo utilizzare la classe wxFileName.Essa espone alcuni metodi per maneggiare file e directory, andiamo a vederne alcuni.

```
#include <wx/filename.h>
wxFileName fname( filename );
if( fname.IsDir() ) return;
wxString path = fname.GetFullPath();
wxString name = fname.GetName();
wxString fullname = fname.GetFullName();
```

Dopo aver istanziato un nuovo oggetto di tipo wxFileName, andiamo a controllare se siamo in presenza di una directory. Possiamo inoltre ottenere il percorso completo del file, oppure il nome completo di estensione. È importante specificare che i metodi su citati non verificano l'esistenza fisica del file. Per verificare che i file esistano fisicamente bisogna utilizzare le funzioni statiche wxFilename::DirExists(path) oppure *wxFilename::FileExists(path)*. Per creare una nuova directory possiamo utilizzare il metodo Mkdir().

LEGGERE E SCRIVERE UN FILE

Andiamo ora a vedere come scrivere un file in wxWidget.

```
wxFile file( "prova.txt", wxFile::write);
if( file.IsOpened() )
{  file.Write("Testo di prova\n");
   file.Write("Linea uno \n");
   file.Write("Linea due \n");
   char data [2] = { 0xFF, 0xC0 };
   file.Write( data , 2 );
   file.Close();
}
```

Inserire testo o dati binari è molto semplice. L'unica accortezza che dobbiamo avere è quella di chiudere il file dopo l'utilizzo. Ora andiamo a vedere come leggere i dati da un file

```
wxFile fileb( "prova.txt", wxFile::read);
if( fileb.IsOpened() )
{  char buffer [ 512 ];
   while( !fileb.Eof() )
  { size_t cnt = fileb.Read(buffer, 511);
     buffer[cnt]= 0;
     printf( buffer ); }
  fileb.Close();
}
```

Se osserviamo la modalità di lettura dei dati ci accorgiamo che la classe wxFile non è altro che un wrapper intorno ai descrittori di file dello stdio. Se abbiamo bisogno di lavorare con file bufferizzati possiamo utilizzare la classe wrapper della struttura FILE wxFFile. La classe wxFile si utilizza allo stesso modo di wxFile. L'unica differenza è la forma di inizializzazione. Infatti al posto di wxFile::read o wxFile::open bisogna usare "r" o "w" o comunque le stringhe di inizializzazione delle funzioni fopen dello standard c. wxWidget mette a disposizione anche gli stream. La gestione degli stream è molto simile alla tradizionale versione del c++ ma con una serie di funzionalità aggiuntive molto utili. Una su tutte è la possibilità di incapsulare tra loro gli stream. Ad esempio creiamo uno stream per la gestione della gestione della scrittura su file:

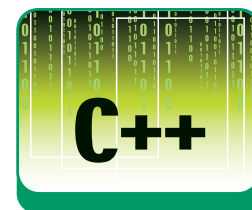
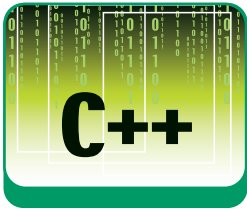
```
wxFFileOutputStream output("mytext.txt");
```

A questo punto potremmo decidere di scrivere sul file nello stile di C++ ed in particolare nel modo in cui scriviamo su cout. Andiamo a creare un nuovo stream che incapsula il precedente e che ci fornisce questa funzionalità:

```
wxTextOutputStream cout( output );
```

A questo punto siamo pronti per scrivere sullo stream.

```
if( output.Ok() )
{ cout << "Testo" << endl;
   cout << 1234 << endl;
   cout << 1.23456 << endl; }
output.Close();
```

dopo aver controllato lo stato dello stream andiamo

**Uno dei problemi ricorrenti delle applicazioni cross-platform è la gestione dei percorsi.
È importante, quindi, utilizzare il meno possibile all'interno del nostro codice riferimenti a file.**

a scrivere alcuni dati ed infine chiudamo l'accesso al file. Chiudendo lo stream più in alto verranno chiusi automaticamente tutti gli stream sottostanti. Nel codice seguente andiamo a leggere i dati appena scritti.

```
wxFileInputStream input( "mytext.txt" );
wxTextInputStream text( input );
  int i1;
  float f2;
  wxString line;
if( input.Ok() )
{ text >> line;
   text >> i1;
   text >> f2;
}
```

Gli stream di input non necessitano della chiusura. Per la lista completa degli stream è bene consultare la documentazione allegata alla libreria.

CROSS -PLATFORM

Con il termine cross-platform vengono indicati i progetti che possono essere ricompilati sotto piattaforme differenti. Scrivere codice completamente portabile è molto complicato poiché ogni sistema operativo mette a disposizione strumenti per lo sviluppo differenti (Win32, gtk, kde, …). Anche la struttura e la dimensione dei tipi di base può cambiare tra compilatori diversi . wxWidgets ci viene incontro mettendoci a disposizione tutti gli oggetti e gli strumenti considerati "critici" evitandoci fastidiosi errori.

UTILIZZARE LE IMMAGINI

Cambiamo decisamente discorso. Ora, ci occuperemo della gestione delle immagini attraverso wxWidgets. Vedremo come caricare, manipolare e salvare i file di immagine. Il caricamento e il salvataggio delle immagini è affidato ad una serie di classi chiamate Image Handlers. Ogni handler si occupa di un formato particolare. Possiamo decidere di caricare un handler in particolare, una serie, oppure tutti quelli disponibili. Per poterne caricare uno basta richiamare la funzione statica wxImage::AddHandler(). Di seguito viene riportato l'elenco completo degli handler disponibili.

```
wxBMPHandler
wxPNGHandler
wxJPEGHandler
wxGIFHandler
wxPCXHandler
wxPNMHandler
wxTIFFHandler
wxIFFHandler
wxXPMHandler
wxICOHandler
wxCURHandler
wxANIHandler
```

Tutti gli handler permettono di caricare e salvare le immagini del formato corrispondente fatta, a eccezione del formato GIF, IFF e ANI. Il wxBMPHandler è sempre caricato e *wxPNGHandler* supporta le trasparenze attraverso il canale alpha. Per caricare tutti gli handler in un colpo solo possiamo utilizzare il metodo wxInitAllImageHandlers(). Ora la comprensione del codice seguente diventa banale.

```
wxInitAllImageHandlers();
wxImage image;
if(image.LoadFile( "image.jpg" ) )
{ int w = image.GetWidth();
  int h = image.GetHeight();
  image.Rescale( w/2 , h/2);
  image = image.Mirror();
  image.Replace( 0x00, 0x00, 0x00, 0x00,0xFF,0x00 );
  image.SaveFile("image.png",wxBITMAP_TYPE_PNG );
}
```

Dopo aver inizializzato tutti gli handler a disposizione proviamo ad aprire un file con la funzione LoadFile. Se tutto è andato bene andiamo ad ottenere le dimensioni dell'immagine. Poi possiamo compiere alcune operazioni sull'oggetto image come la riscalatura, il ribaltamento, e la sostituzione del colore nero(0x000000) con il verde (0x00FF00); poi scegliamo il file di salvataggio e salviamo il lavoro come file PNG. La funzione SaveFile ci mette a disposizione due modi di scegliere il formato di salvataggi. Il primo prevede l'identificazione del formato dall'estensione del nome del file (questo metodo risulta efficace quando si utilizzano le estensioni standard) oppure tramite il passaggio di un parametro che identifica il formato; in questo caso è possibile scegliere un'estensione del file non standard. Le costanti che identificano i formati delle immagini sono le seguenti :

```
wxBITMAP_TYPE_BMP
wxBITMAP_TYPE_PNG wxBITMAP_TYPE_JPEG
wxBITMAP_TYPE_PCX
wxBITMAP_TYPE_PNM
wxBITMAP_TYPE_TIFF
wxBITMAP_TYPE_XPM
wxBITMAP_TYPE_CUR
wxBITMAP_TYPE_ICO
```

queste costanti sono utilizzabili anche con il metodo LoadFile per forzare il riconoscimento del tipo di file.

I DEVICE CONTEXT

I device Context (DC) rappresentano una serie di classi che possono essere utilizzate per le operazioni di disegno su una periferica supportata. Attraverso i DC è possibile disegnare sullo schermo intero, su una finestra, sulla stampante, su un file immagine o su un documento PostScript utilizzando sempre la stessa sintassi. Infatti tutti i device contex citati ereditano dalla classe wxDC. Va ricordato che anche in questo caso il codice scritto è valido su tutte le piattaforme supportate da wxWidgets. Nell'esempio seguente realizzeremo un'immagine da zero e poi la salveremo come un file png.

```
wxBitmap bmp( 200, 200 );
wxMemoryDC dc;
dc.SelectObject( bmp );
```

Istanziamo un oggetto di tipo wxBitmap e diamogli la dimensione quadrata 200x200. Poi Realizziamo un DC virtuale in memoria e gli associamo la bitmap. Poi trattiamo la variabile dc come un DC tradizionale e procediamo con le operazioni di disegno.

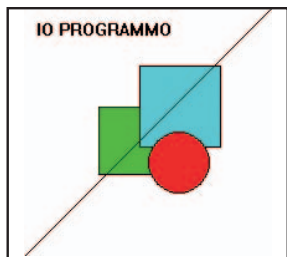



Fig. 1: ***Il risultato dell'uso di una wxBitmap***

```
dc.SetBackground( *wxWHITE_BRUSH );
dc.Clear();
dc.DrawText(_T("IO PROGRAMMO"),10,10 );
dc.SetPen( *wxBLACK_PEN);
dc.SetBrush( *wxGREEN_BRUSH );
dc.DrawRectangle( 60, 80 , 65 , 55 );
dc.SetBrush( *wxCYAN_BRUSH );
dc.DrawRectangle( 93, 47 , 66 , 66 );
dc.SetBrush( *wxRED_BRUSH );
dc.DrawEllipse( 100, 100 , 50 , 50 );
dc.DrawLine( 200, 0 , 0 , 200 );
dc.SelectObject( wxNullBitmap );
bmp.SaveFile("image.png",  wxBITMAP_TYPE_PNG);
```

Il risultato di queste operazioni è visibile nella **Figura 1**.

GESTIONE DEGLI APPUNTI

Una delle funzionalità più utili, che un programma possa mettere a disposizione, è senza dubbio la gestione degli appunti o clipboard. Per capirci meglio la possibilità di utilizzare il taglia,copia e incolla. WxWidgets si occupa di fornire al programmatore una semplice soluzione per l'utilizzo di questa caratteristica sempre in modo cross-platform. Questo viene fatto mettendo a disposizione un oggetto globale wxTheClipboard disponibile inserendo la direttiva #include <wx/clipbrd.h>. Per utilizzare la clipboard bisogna innanzitutto ottenerne il possesso attraverso il metodo Open poi possiamo utilizzarla. Vediamo ora come inserire del testo negli appunti.

```
if (wxTheClipboard->Open())
{  wxTheClipboard->SetData
     (new wxTextDataObject("Testo Salvato"));
     wxTheClipboard->Close(); }
```

possiamo anche salvare un'immagine...

```
if (wxTheClipboard->Open())
{ wxTheClipboard->SetData
   (new wxBitmapDataObject(bitmap));
   wxTheClipboard->Close();
}
```

Per leggere il testo degli appunti dobbiamo ottenere l'escusiva sulla clipboard controllare se i dati in memoria sono di tipo testuale e poi procedere con l'estrazione...

```
if (wxTheClipboard->Open())
{ if
    (wxTheClipboard->IsSupported(wxDF_TEXT))
    {  wxTextDataObject data;
        wxTheClipboard->GetData( data );
        wxMessageBox( data.GetText() ); }
   wxTheClipboard->Close();
}
```

Anche questa operazione è molto semplice da realizzare ma allo stesso tempo molto utile.

USARE I WXSOCKET

Tra le tante classi troviamo anche il supporto ai socket. In wxWidget, a differenza degli strumenti proposti nativamente dai sistemi operativi, è molto semplice operare con questi costrutti. A nostra disposizione vengono messe a disposizione due classi wxSocketServer e wxSocketClient. La prima si occupa di creare un servizio e mettersi in ascolto su una porta la seconda invece ci semplifica la vita qualora noi volessimo connetterci ad un servizio in ascolto su una porta. Per creare un servizio bisogna innanzitutto definire la porta; questo viene fatto attraverso la classe wxIPV4address che definisce gli indirizzi attraverso il protocollo IPV4, poi scegliamo la porta del servizio stesso.

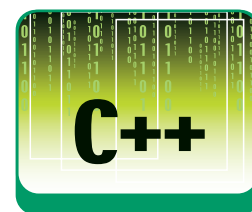
```
wxIPV4address addr;
   addr.Service(3000);
```

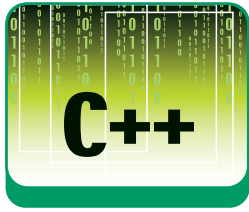
Il codice appena visto corrisponde all'indirizzo "localhost:3000". ora siamo pronti per creare il servizio

```
wxSocketServer * m_server = new wxSocketServer(addr);
```

A questo punto abbiamo creato il socket client e dobbiamo metterlo in ascolto. Avviamo il servizio in modo sincrono, ovvero il socket bloccherà l'esecuzione del programma sino alla prima richiesta di connessione

```
wxSocketBase * client = m_server->Accept(true);
```

Appena sarà disponibile una richiesta, la variabile

Il progetto wxWidgets è disponibile all'indirizzo
http://www.wxwidgets.org

Tutte le informazioni su wxDevCpp sono reperibili al seguente indirizzo.
http://wxdsgn.sourceforge.net/

client punterà ad un'oggetto di tipo wxSocketBase se nessuna richiesta giungerà al servizio per una durata di 10 minuti (timeout) la funzione Accept() restituirà il valore NULL. Il valore true passato alla funzione Accept() indica che tale funzione aspetterà la durata del timeout o l'arrivo di una richiesta di connessione prima di restituire un valore. Per variare il tempo di timeout possiamo utilizzare la funzione SetTimeout(int seconds) quindi selezionare il nuovo tempo massimo di attesa. Possiamo evitare lo stallo del programma ed utilizzare la versione non bloccante di Accept().

NOTA

wxDevcpp installa una serie di packages predefiniti tra i quali wxWidgets 2.6.1 completo di guida in linea.

```
 wxSocketBase * client = NULL;
while( client == NULL )
{ //... Fai altre operazioni
  client = m_server->Accept(false);
}
```

Una volta ottenuto un client in connessione possiamo procedere con la comunicazione. Definiamo un piccolo protocollo di comunicazione per ricevere stringhe di testo. Il cliente appena collegato invia al server un comando che identifica l'operazione da compiere. Il comando viene inviato con un char, ed in particolare decidiamo di inviare 0xBE. A questo punto andiamo ad aspettare la definizione della lunghezza della stringa. Ipotizziamo che le stringhe siano lunghe al massimo 255 caratteri quindi utilizziamo un unsigned char per ricevere la dimensione del testo, prepariamo un buffer per ricevere i dati e riceviamo la stringa. A questo punto inviamo la stringa ricevuta al mittente come test.

```
unsigned char c;
 client->Read(&c, 1);
 if( c ==  0xBE )
 { unsigned char len;
    char *buf;
    client->Read(&len,1);
    buf = new char[len];
    client->Read(buf, len);
    client->Write(buf, len);
    delete[] buf; }
 else
 { printf("Comando errato");}
delete m_server;
```

Per interrompere il servizio basta eliminare l'oggetto che lo realizza, nel nostro caso m_server. Andiamo ora a realizzare il client. Il codice del client va scritto e compilato in progetto differente dal codice del server. Inoltre, è semplice intuirlo, il client può girare anche su una macchina differente da quella in cui viene eseguito il server. Prepariamo l'indirizzo :

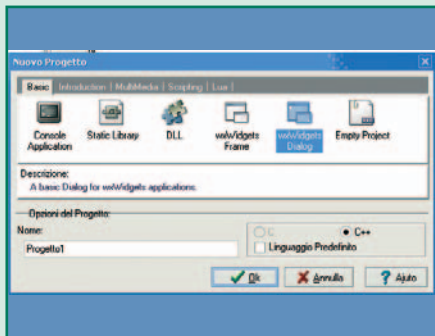
```
 wxIPV4address addr;
 addr.Hostname("127.0.0.1");
 addr.Service(3000);
```

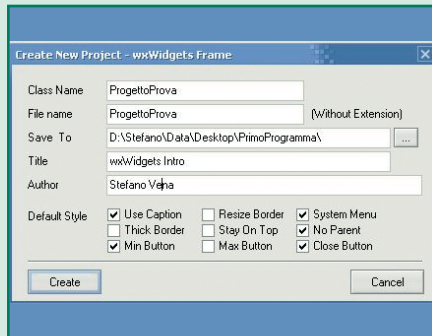
Scegliamo come indirizzo IP il localhost e come porta la 3000, ovvero scegliamo l'indirizzo e la porta del servizio creato in precedenza, poi proviamo a stabilire una connessione:

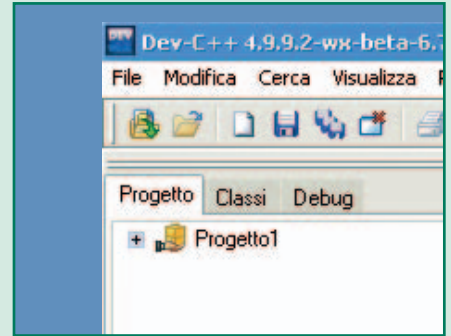
```
wxSocketClient * m_sock = new wxSocketClient();
client->Connect(addr, false);
```

IL MIO PRIMO PROGETTO CON WXDEVCPP

Vedremo in sei semplici passi come sfruttare un DEV C++ come IDE RAD per lo sviluppo di interfacce grafiche. Scopriremo che il m eccanismo è identico a quello degli IDE più evoluti

> UN NUOVO PROGETTO

1 Una volta eseguito wxDevcpp selezioniamo dal menu File la voce Nuovo->Progetto. Dalla sezione basic scegliamo di creare un progetto dal template "wxWidgets Dialogs" inseriamo un nome per il nostro lavoro e diamo l'ok!

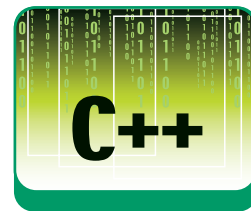
> QUALI CLASSI?

2 Il prossimo passaggio è definire il nome dei file e delle classi che compongono il progetto di base. La procedura è guidata. Una volta completate le procedure di configurazione siamo pronti per iniziare a lavorare.

> I FILE DELLE FORM

3 wxDevcpp salva le informazioni sui dialoghi grafici in file di estensione wxform. È sufficiente andare nel visualizzatore dei file del progetto e selezionare i file con tale estensione per accedere all'editor di dialoghi.

```
bool aspetta = true;
while( !client->WaitOnConnect(10,0) &&  aspetta )
{  //..possiamo dare informazioni
   //all'utente oppure aggiornare lo
   //stato di aspetta}
bool ok = client->IsConnected();
```

Se tutto è andato a buon fine e il flag ok è vero, possiamo iniziare il dialogo con il servizio.

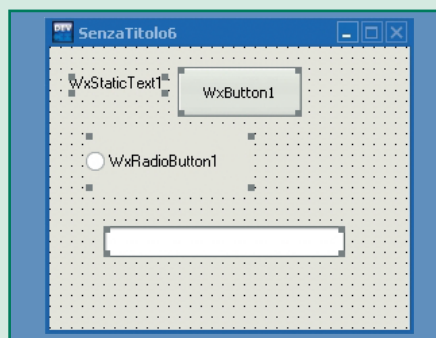
```
if(ok)
{ const wxChar *buf1;
  wxChar       *buf2;
  unsigned char len;
  unsigned char c = 0xBE;
  client->Write(&c, 1);
  buf1 = _("Testo di prova più corto di 256 caratteri!)");
  len  = (unsigned char)
     (wxStrlen(buf1)+1)*sizeof(wxChar));
  buf2 = new wxChar[wxStrlen(buf1) + 1];
  printf("Inviamo il testo al server...\n");
  client->Write(&len, 1);
  client->Write(buf1, len);
  printf( client->Error() ? "Errore!\n" : "Fatto\n" );
  printf("Riceviamo la risposta dal server ...");
  client->Read(buf2, len);
  printf( client->Error() ? "Errore!\n" : "Fatto\n" );
  printf(_("Compariamo i buffer..."));
  if (memcmp(buf1, buf2, len) != 0)
  { printf("Errore nella comunicazione!\n");}
  else
  {  printf("Comunicazione andata a buon fine!\n");}
  delete[] buf2; }
```

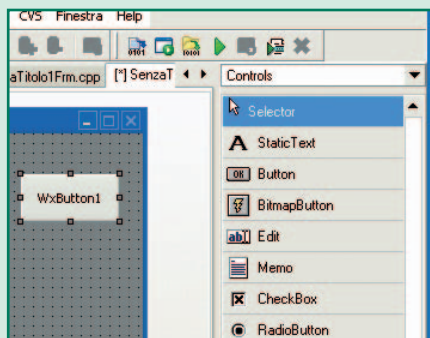
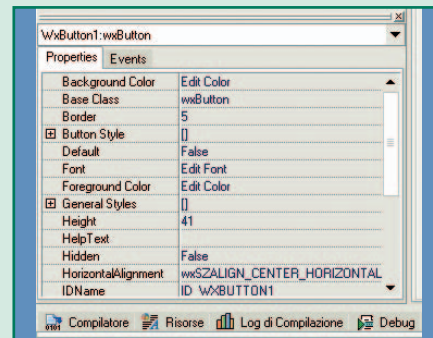
Purtroppo non ci sono modi di controllare se i dati inviati corrispondono alla dimensione attesa. Nel caso in cui i dati inviati risultino essere minori di quelli attesi è possibile che si verifichino errori di overflow mentre nel caso in cui la dimensione dei dati comunicata sia inferiore di quella effettivamente inviata, i dati ricevuti saranno errati.

GESTIONE DEI THREAD

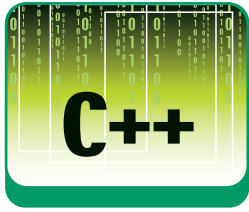
Una aspetto della programmazione che non poteva mancare in questo pacchetto è la gestione dei thread. In wxWidgets la gestione dei thread o processi leggeri è realizzata utilizzando il paradigma della programmazione ad oggetti. Infatti la libreria mette a disposizione una classe astratta, wxThread, dalla quale ereditare le classi che contengono il codice da eseguire all'interno di un thread secondario. Sono messi a disposizione dello sviluppatore anche una serie di oggetti che riguardano la programmazione concorrente in generale. Questi oggetti sono wxMutex e wxCriticalSection per gestire la mutua esclusione, wxCondition per implementare meccanismi di sincronizzazione mediante eventi, wxSemaphore per la sincronizzazione attraverso il paradigma dei semafori. Per prima cosa vediamo come avviare un nuovo thread. La prima cosa da fare è usare la direttiva "#include <wx/thread.h>" successivamente è possibile creare la classe che incorpora il thread. Come esempio vediamo la realizzazione di un thread che scrive su di una textbox in modo asincrono al flusso principale della nostra applicazione.

THREADING

Il multithreading è una caratteristica dei sistemi operativi molto interessante. Ma essere in grado di far girare il codice in modo parallelo è allo stesso tempo affascinante e pericoloso. Non accade così di rado che un'applicazione vada in stallo per una gestione scorretta della sincronia tra thread differenti. A volte l'intero sistema si blocca. Per fortuna *wxWidget* ci viene incontro con una serie di strumenti molto utili. Ma la riuscita di una applicazione dipende sempre dalla accuratezza impiegata dallo sviluppatore durante la realizzazzione.

› DISEGNIAMO

4 In questa modalità è possibile inserire nel dialogo tutti gli oggetti messi a disposizione dal framework. Lo stile è quello classico degli editor RAD, è sufficiente trascinare il controllo desiderato sulla form per ottenerne una riproduzione fedele

› DOVE SONO I CONTROLLI ?

5 Tutti i controlli (e non solo) inseribili nei frame sono elencati nella barra a destra dell' IDE. E' possibile scegliere fra bottoni, label, combo, tutti i classici controlli disponibili, ovviamente wxWidgets funzionerà sia su sistemi Windows che Linux

› LE PROPRIETÀ

6 Per ogni controllo selezionato è possibile modificarne i parametri attraverso l'utile editor di proprietà in stile VisualStudio. Anche in questo caso il meccanismo è quello tradizionale degli editor RAD evoluti. Ogni controllo possiede proprietà e metodi

```
class MyThread : public wxThread
{ public:
    MyThread(MyFrame *frame);
    virtual void *Entry();
    virtual void OnExit();
    void WriteText(const wxString& text);
 public:
    size_t   m_count;
    MyFrame *m_frame;
};
```

La derivazione dalla classe astratta wxThread impone di implementare il metodo _virtual void* Entry()" questo metodo è il cuore del processo. Infatti una volta creato il thread viene passato il controllo alla funzione Entry. All'uscita dalla procedura il thread viene interrotto. Andiamo a vedere in dettaglio i vari passaggi.

```
MyThread::MyThread(MyFrame *frame): wxThread()
{  m_count = 0;
   m_frame = frame;
}
```

Il costruttore prende come parametro il frame contenente il textbox sul quale andare a scrivere.

```
void MyThread::WriteText (const wxString& text)
{  wxString msg;
    wxMutexGuiEnter();
    msg << text;
    m_frame->WriteText(msg);
    wxMutexGuiLeave();
}
```

La scrittura sul textbox è un procedimento delicato che richiede la sincronizzazione. Quindi possiamo gestire la mutua esclusione sulle operazioni di aggiornamento dell'interfaccia utente attraverso la chiamata al metodo wxMutexGuiEnter(). Appena ottenuto il permesso alla scrittura, compiamo l'operazione e rilasciamo la risorsa attraverso wxMutexGuiLeave().

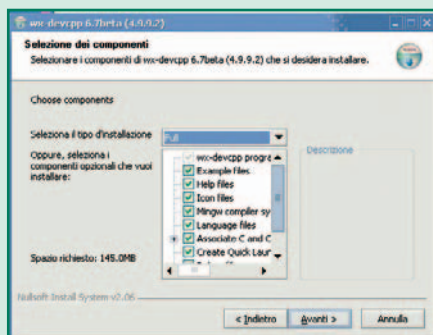
```
void MyThread::OnExit()
{ }
```

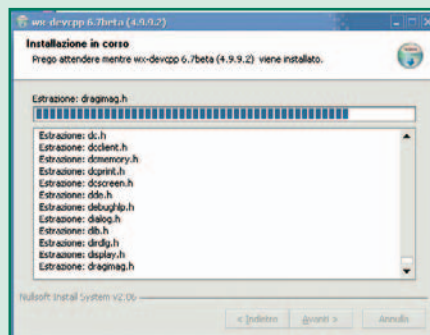
Quando le operazioni del thread terminano, oppure quando si decide di interrompere il procedimento, viene richiamata la funzione OnExit(). Nel nostro caso non sono necessarie operazioni "post-thread"; quindi la nostra funzione è vuota. Andiamo ora a vedere il nucleo del thread.

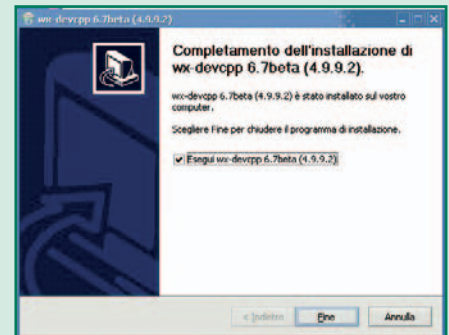
```
void *MyThread::Entry()
{ wxString text;
   text.Printf( wxT("Thread 0x%lx avviato(priorità =
                        %u).\n"), GetId(), GetPriority());
   WriteText(text);
  for (m_count = 0;
        m_count < 10;
       m_count++ )
{ if ( TestDestroy() ) break;
     text.Printf(wxT("[%u] Thread 0x%lx Scrive.\n"),
                                    m_count, GetId());
        WriteText(text);
        wxThread::Sleep(1000);}
    text.Printf(wxT("Thread 0x%lx è terminato.\n"),
                                               GetId());
    WriteText(text);
    return NULL;
}
```

INSTALLARE WXDEVCPP

Il processo di installazione è sufficientemente semplice. L'intera procedura è guidata da un comodo wizard. Seguiamolo passo passo e scegliamo le opzioni più utili

> IL VIA

1 Il processo di installazione è molto semplice! Basta eseguire il setup ed il gioco è fatto!

> IL COMPILATORE

2 Il Setup installa oltre all'IDE la versione 3.4.0 del compilatore gcc più le librerie statiche di wxWidgets

> DEV C++

3 Non appena l'installazione è completata scegliamo di avviare DevC++ per configurarlo.

Il codice mostrato è molto semplice; bisogna, però, fare attenzione ad una serie di fattori. È buona norma controllare spesso, all'interno della funzione Entry il valore restituito dalla funzione TestDestroy(). Se il valore restituito è true è necessario uscire dal thread il prima possibile. TestDestroy() restituisce true in caso di errori oppure nel caso in cui l'utente decida di interrompere il thread prima della sua naturale conclusione. A questo punto avviare il thread è semplice

```
MyThread *thread = new MyThread(this);
if(thread->Create()!= wxTHREAD_NO_ERROR )
{ wxMessageBox(wxT("Impossibile creare il thread!"));}
else
    thread->Run();
```

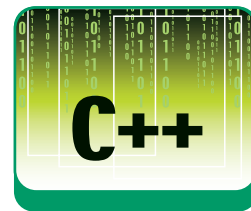
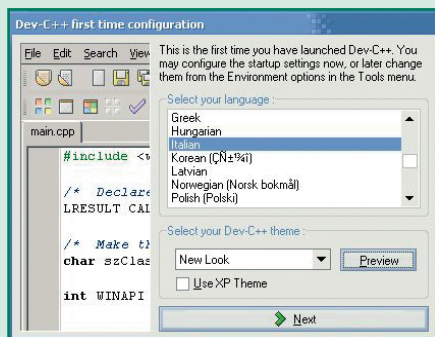
Per metterlo in pausa possiamo utilizzare la funzione thread->Pause(). Su alcuni sistemi questo comando ha efficacia immediata, ovvero il thread viene interrotto subito. Su altri sistemi invece il flusso si interrompe dopo la prima invocazione di TestDestroy(). Per ripristinare un thread in pausa basta richiamare thread->Resume(). Per interrompere il flusso invece è necessario chiamare thread->Delete(). Questa chiamata non solo interrompe l'esecuzione del thread ma rilascia anche le risorse ad esso associate e distrugge l'oggetto. È inutile andare a spiegare in dettaglio quali sono i problemi legati alla sincronizzazione, perché noti alla grande maggioranza dei lettori e poi l'argomento è stato trattato in modo specifico negli scorsi numeri di io Programmo. Per limitare la possibilità di errore dello sviluppatore, wxWidgets mette a disposizione una serie di classi wrapper il cui scopo è quello di gestire lucchetti e sezioni critiche. Queste classi sono wxMutexLocker, wxCriticalSectionLocker. Il costruttore della classe wxMutexLocker bloccare un oggetto mutex. Tale oggetto viene poi rilasciato dal distruttore. Lo stesso vale per un oggetto di tipo wxCriticalsectionLocker. All'atto della costruzione avviene l'ingresso in una sezione critica. L'uscita avviene nel momento in cui viene invocato il distruttore. Di seguito viene riportato un esempio di tale operazione.

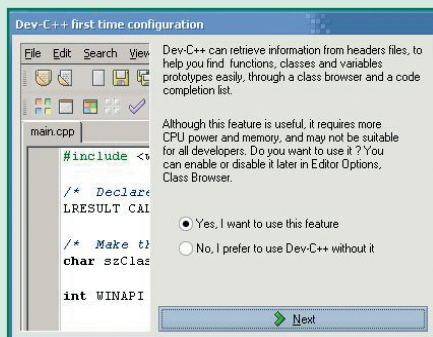
```
void Funzione_di_prova()
{  wxCriticalSectionLocker locker ( sezione_critica);
    if ( ... )
    {  //compi delle operazioni
        return; }
   //fai altre operazioni...
    return;
}
```

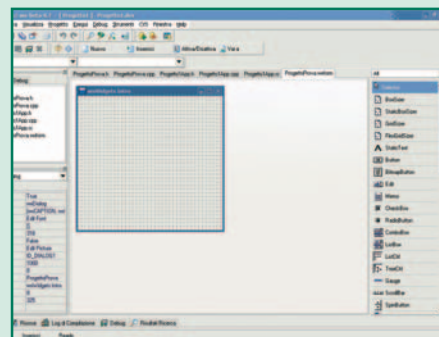
subito dopo aver richiamato return; Il distruttore dell'oggetto locker si occupa di uscire dalla sezione critica individuata dall'oggetto globale sezione_critica.

CONCLUSIONI

Le possibilità offerte dal framework wxWidgets sono vaste e risulta difficoltoso riassumerle in poco spazio. La libreria comunque è documentata molto bene e non mancheremo di trattare ulteriormente le funzioni più interessanti

Stefano Vena

> QUALE LINGUA?

4 Nella prima fase del setup possiamo scegliere la lingua e lo stile dell'interfaccia utente.

> SUPER EDITOR

5 Poi possiamo scegliere se abilitare o meno il completamento automatico del codice e il navigatore di classi.

> L'ARRIVO

6 Ora siamo pronti per utilizzare il nostro IDE gratuito integrato con wxWidgets.

MANDARE UN SMS O UN MMS A TEMPO

LE WIRELESS MESSAGING API CI CONSENTONO DI REALIZZARE MIDLET PER INVIARE SMS E MMS. VEDREMO COME FARE IN MODO CHE I MESSAGGI VENGANO SPEDITI QUANDO DESIDERIAMO

Tra i package opzionali che J2ME mette a disposizione degli sviluppatori, uno dei più interessanti è il WMA *(Wireless Messaging API)*. Questo insieme di API consente, nella versione 1.1, di inviare SMS *(Short Message Service)* e CBS *(Cell Broadcast short message Service)*, nella versione 2.0 di inviare anche MMS *(Multimedia Message Service)*.
In quest'articolo vedremo prima di tutto come realizzare una MIDlet per inviare SMS e come fare in modo che il messaggio venga spedito un certo giorno ad una certa ora. Ci adopereremo poi a superare un ostacolo che tutti gli sviluppatori J2ME hanno certamente incontrato: quello dei tanto odiati "permessi" che, se da una parte sono sicuramente uno strumento per la sicurezza, dall'altro costituiscono un momento spesso fastidioso per un utente che utilizza applicazioni nelle quali viene utilizzata una classe considerata non sicura.
In particolare per l'idea che vogliamo mettere in pratica, la problematica dei permessi rende poco utile l'applicazione. Supponiamo infatti di voler utilizzare un'applicazione di *"scheduled messaging"*, perché dimentichiamo di fare gli auguri ai nostri amici. Quindi, componiamo il messaggio di auguri una settimana prima (o quando ci viene in mente) e lasciare che la nostra applicazione lo invii all'occorrenza può essere un'ottima soluzione. In questo caso, se il giorno del compleanno del nostro amico, all'ora da noi decisa, il programmino ci chiederà *"Sei sicuro di voler inviare il messaggio?"*, la cosa può anche non disturbarci ma, sicuramente, a parte forse il tempo risparmiato per pensare e scrivere il messaggio, rispetto ad un pro-

COME FUNZIONERÀ L'APPLICAZIONE?

Vediamo i semplici passi che consentono di schedulare l'invio di SMS

› START SMS!

1 Avviata l'applicazione per l'invio degli SMS, compare la schermata dove inserire numero del destinatario, data e ora di spedizione.
Abbiamo aggiunto qualche controllo per la validità dei formati.

› SCRITTURA E INVIO SMS

2 Premendo il tasto relativo allo voce *"OK"* dalla schermata iniziale, compare la schermata dove scrivere il testo del messaggio. Premendo *"Invia"* l'applicazione si chiude e alla data e ora stabilita l'SMS verrà inviato.

› START MMS!

3 Avviata l'applicazione per l'invio degli MMS, compare la schermata dove inserire subject, destinatario, data e ora di spedizione.
Anche in questo caso valgono i controlli usati nel passo 1.

gramma agenda in cui si segnano gli appuntamenti e che produce un alert all'ora desiderata, non è cambiato granché.
Ricordiamo a chi non fosse un grande esperto che anche settando nel jad relativo alla MIDlet i permessi per una determinata classe, ovvero la possibilità che la classe venga eseguita senza richiederne conferma all'utente, in realtà ha validità solo se la nostra applicazione risulta *"signed"*, ovvero certificata da un apposito ente e si connette soltanto a domini protetti (*Trusted 3rd party*). Ovviamente far certificare la propria applicazione costa… Quello che faremo per aggirare il problema sarà una sorta di hacking, ovvero cloneremo il protocollo sms con un altro protocollo (*"vsms"*) per il quale non setteremo alcuna opzione relativamente ai permessi. Fatto ciò vedremo come realizzare una MIDlet per inviare MMS in maniera programmata. Questa volta lasceremo a voi lettori la possibilità di realizzare l'hacking, del tutto simile a quello del caso precedente.

INVIO SCHEDULATO DEGLI SMS

La MIDlet che utilizziamo per la programmazione e gli invii di SMS è *SMSMIDlet*.
Progettiamo questa MIDlet in modo che quando avviamo l'applicazione ci compaia un Form contenente un *TextField* e un *DateField*.
Nel *TextField* l'utente può inserire il numero di telefono del destinatario. Alcuni telefoni, per default, consentiranno di aggiungere il numero dalla rubrica. Nel *DateField* viene visualizzata l'ora e la data attuale. Se l'utente non modifica tale impostazione l'SMS sarà spedito immediatamente, altrimenti sarà inviato alla data e ora di spedizione.
Vediamo come costruire questi due campi e come inserirli nel Form:

```
Form smsForm = new Form("SMS");
TextField destinationAddressField = new
    TextField("A:",null, 13, TextField.PHONENUMBER);
DateField sendingDateField = new DateField("Ora
    e data ora di spedizione:", DateField.DATE_TIME);
Date actualDate = new Date();
sendingDateField.setDate(actualDate);
smsForm.append(destinationAddressField);
smsForm.append(sendingDateField);
```

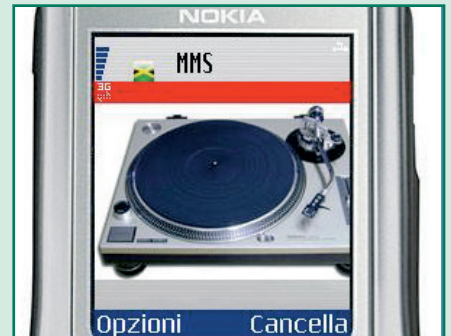
Come si vede, nella costruzione del *TextField* specifichiamo che il testo sia un numero di telefono. Inoltre con il metodo *setDate di DateField* impostiamo come data da visualizzare quella attuale. Una volta che l'utente ha inserito queste informazioni non gli resta che scrivere il messaggio. Il *Form* avrà pertanto un *Command* che gli consente di visualizzare un'ulteriore schermata. Prima che ciò avvenga possiamo pensare di eseguire un controllo sul numero di telefono, per stabilire se sia corretto. Infatti, non aver specificato come tipo di testo un numero di telefono, non ci preserva dall'inserimento di un numero non corretto. Il campo *PHONENUMBER* prevede, oltre ai numeri, un insieme di caratteri, che variano da telefono a telefono, e

> SCELTA TIPO ALLEGATO

4 Premendo il tasto relativo allo voce "Inserisci oggetto" dalla schermata iniziale, compare la lista delle tipologie di allegato. Bisogna selezionare la tipologia desiderata e premere *"Avanti"*. Scegliendo *"Testo"* si passa alla scrittura del testo.

> SELEZIONE FILE ALLEGATO

5 Scegliendo le opzioni diverse da *"Testo"* si apre un file browser. Scorrendo tra le directory si arriva a selezionare il file desiderato premendo *"Apri/Seleziona"*. Esattamente come quando alleghiamo un file in attach ad una e-mail

> VISUALIZZAZIONE ALLEGATO E INVIO MMS

6 Il file viene aperto e potete vedere l'immagine o il video oppure ascoltare l'audio. Premendo *"OK"* tornate alla schermata iniziale, premete *"Invia"*. L'applicazione si chiude e alla data e ora stabilita l'MMS verrà spedito.

che sono ad esempio "+" e "*". Un numero del tipo *"++393…"* ad esempio non è corretto. Realizziamo allora una funzione che controlla che la lunghezza del numero sia maggiore di 0, e che accetti numeri che iniziano con "+":

```
private static boolean isValidPhoneNumber(String
                                         number) {
[...]
}
```

In realtà possiamo eseguire altri controlli, anche sulla lunghezza, che abbiamo deciso essere al massimo di 13 caratteri.
Quando l'utente digiterà sul telefono il tasto corrispondente ad *"OK"* verrà chiamato il metodo *preSending( )*:

```
private void preSending () {
    [...]
      smsSending.composeSMS("sms://" + address,
                                      sendingDate);
  }
```

Trovate l'implementazione completa nel CD. Questo metodo verifica la correttezza del numero, in caso negativo visualizza una pop-up di errore (utilizzando un *Alert*) e infine chiama il metodo *composeSMS* dell'oggetto *smsSending* di tipo *SMSsending*. La classe *SMSsending* è quella che si occupa dell'effettivo invio del messaggio. Essa estende la classe *TimerTask* responsabile dell'esecuzione schedulata. Il costruttore di *SMSsending* è il seguente:

```
  public SMSsending(Display display,
     Displayable backScreen, SMSMIDlet smsMIDlet) {
      this.display = display;
      this.destinationAddress = null;
      this.backScreen = backScreen;
      this.smsMIDlet = smsMIDlet;
      messageAlert = new Alert("", null, null,
                                  AlertType.ERROR);
      messageAlert.setTimeout(5000);
      messageBox = new TextBox("Testo Messaggio",
                        null, 65535, TextField.ANY);
      messageBox.addCommand(backCommand);
      messageBox.addCommand(sendCommand);
      messageBox.setCommandListener(this);
  }
```

La classe in questione prende dalla MIDlet i parametri di cui ha bisogno, ovvero il riferimento al *Display*, il *Form* di start a cui si deve poter ritornare e un'istanza della MIDlet. Inoltre viene costruita la *TextBox* da utilizzare per la scrittura del messaggio.
Il metodo *composeSMS* invocato dalla MIDlet non fa altro che acquisire l'indirizzo di destinazione del messaggio e la data di invio, e visualizza la *TextBox*:

```
public void composeSMS(String destinationAddress,
                             Date sendingDate) {
      this.destinationAddress = destinationAddress;
      this.sendingDate = sendingDate;
      display.setCurrent(messageBox);
  }
```

Si noti che l'indirizzo di destinazione è del tipo *protocollo://indirizzo:porta*, nel nostro caso *"sms://393…"*. Non specificando la porta viene utilizzata quella assegnata per default dal telefono all'invio degli SMS.
Una volta che l'utente ha terminato la scrittura del messaggio utilizzerà il tasto *"Invia"* per spedirlo. Nel *CommandAction* la pressione di questo tasto genera questo effetto:

```
if (c = = sendCommand) {
            Timer sendingTimer = new Timer();
            TimerTask sendingTask = this;
sendingTimer.schedule(sendingTask,sendingDate);
            smsMIDlet.sleepApp();
        }
```

In pratica costruiamo un *Timer*, e un *TimerTask*, ovvero l'azione da eseguire quando il *Timer* scade, e scheduliamo il *task* alla data e ora decisa dall'utente. Siccome la classe *SMSsending* estende *TimerTask*, il *task* eseguito allo scadere del *Timer* è il metodo *run ()* di *SMSsending*.

FILE CONNECTION API

Le *FileConnection API* insieme alle PIM *(Personal Information Management)* API costituiscono il *PDA Profile.* Si tratta di funzionalità per poter, rispettivamente, accedere al file system del telefono e di eventuali memorie esterne e alle informazioni personali quali quelle contenute nella rubrica, nell'agenda, nelle note. È solo da poco tempo che i telefoni cellulari hanno iniziato a fornire il supporto per questo tipo di API. E fino ad allora ciò rendeva le applicazioni scritte in codice nativo su sistema operativo (Symbian OS su tutti) inarrivabili rispetto alle applicazioni J2ME, relegate spesso nel contesto del mobile *gaming.* Ora finalmente si possono sviluppare applicazioni utili e interessanti, sfruttando i vantaggi di Java: portabilità e compatibilià. Le *FileConnection API* sono definite nel package *javax.microedition.io.file* e permettono, come già detto di accedere al file system, in lettura e in scrittura. Queste sono le interfacce/classi comprese:

- **FileSystemRegistry: classe che permette di sapere quali file system sono presenti sul nostro dispositivo (memoria interna, memory card, ecc.)**
- **FileSystemListener: interfaccia utilizzata per avere informazioni sui cambiamenti del file system (ad esempio inserimento di memory card)**
- **FileConnection: interfaccia per la connessione a file e directory in lettura e in scrittura**
- **ConnectionClosedException: eccezione generata quando si cerca di effettuare un'azione su un file ma la connessione risulta chiusa**
- **IllegalModeException: eccezione generata quando si cerca di accedere ad un file in modalità diversa rispetto a quella utilizzata in apertura.**

Nel frattempo che arriva il momento dell'invio la MIDlet viene resa "dormiente":

```
public void sleepApp() {
    display.setCurrent(null);
}
```

Ciò significa che l'applicazione resta aperta, non rilascia le sue risorse, pur se "visivamente" sembra di esserne usciti.
Come abbiamo detto, allo scadere del *Timer* viene eseguito il metodo *run()*:

```
public void run() {
    String address = destinationAddress;
    MessageConnection smsconn = null;
    try {
        smsconn = (MessageConnection)
                            Connector.open(address);
        TextMessage txtmessage = (TextMessage)
                            smsconn.newMessage(
                    MessageConnection.TEXT_MESSAGE);
        txtmessage.setPayloadText(
                            messageBox.getString());
        smsconn.send(txtmessage);
    } catch (Throwable t) {
        messageAlert = new Alert("", "SMS non
                    inviato!", null, AlertType.ERROR);
        messageAlert.setTimeout(5000);
        display.setCurrent(messageAlert);
        smsMIDlet.quitApp();
    }
    if (smsconn != null) {
        try {
        smsconn.close();
        messageAlert = new Alert("", "SMS
                    inviato!", null, AlertType.INFO);
            messageAlert.setTimeout(5000);
                display.setCurrent(messageAlert);
                smsMIDlet.quitApp();
        } catch (IOException ioe) {
            messageAlert = new Alert("",
                    "Connessione non riuscita!",
                        null, AlertType.ERROR);
            messageAlert.setTimeout(5000);
                display.setCurrent(messageAlert);
                smsMIDlet.quitApp();
        }
    }
}
```

Analizziamo il comportamento di questo metodo. La prima cosa che ci si aspetta è che venga stabilita una connessione. E infatti l'invio di sms è implementato da un'interfaccia *Connection* e, nel caso specifico, da un *MessageConnection.*. Per aprire la connessione, l'applicazione ottiene un oggetto che implementa il *MessageConnection* dalla classe *Connector*, attraverso una URL che identifica l'indirizzo. Per quanto riguarda il messaggio vero e proprio, esso è rappresentato dall'interfaccia *Message* e dalle interfacce da essa derivate. In particolare per un SMS, l'interfaccia a cui far riferimento è *TextMessage*. Si costruisce quindi un nuovo *TextMessage* a partire dal *MessageConnection* e poi si "riempie" il suo payload con la stringa di testo ottenuta dal *MessageBox*. A questo punto utilizzando il metodo send di *MessageConnection* il messaggio viene spedito.
Da notare che i protocolli wireless a cui si accede tramite le WMA sono tipicamente del tipo store-and-forward, diversamente dai datagram del livello rete. Questo significa che il messaggio viene consegnato quando il destinatario sarà connesso, anche se questo non accade al momento dell'invio.
Per rendere maggiormente usabile la nostra applicazione ricorriamo ad una serie di *Alert* che possano comunicare all'utente, se il messaggio è stato inviato o meno o se, non essendoci campo, non è stata possibile stabilire la connessione. Si ricordi che questo è il caso di invio "standard", cioè quello in cui al momento dell'invio il telefono chiede all'utente il permesso di inviare il messaggio.

HACKING DEL PROTOCOLLO "SMS"

Il nostro scopo è quello di rendere l'applicazione completamente autonoma, e dover confermare l'invio del messaggio risulta in questo senso veramente fastidioso.
Come già accennato utilizziamo allora un "clone" del protocollo *"sms"*. Lo chiamiamo *"vsms"* ma potete chiamarlo come più vi piace.
Le classi che abbiamo descritto finora restano inalterate tranne per il fatto che nell'indirizzo di destinazione il nome del protocollo ora sarà *"vsms"* e non più *"sms"*.
Le WMA implementano il protocollo sms attraverso una serie di classi che, a seconda del telefono, sono contenute un diverso package tra quelli che costituiscono il MIDP. Ad esempio per la Series60 di Nokia, che ha un sistema operativo Symbian, il package in questione è il seguente: *com.symbian.midp.io.protocol.sms*.
La realizzazione del protocollo "vsms", nascerà quindi dalla creazione del package *com.symbian.midp.io.protocol.vsms*.
All'interno di questo package scriviamo due classi, presenti anche nel package *sms: Protocol e SMSClientConnectionImpl*. In realtà nel package *sms* è contenuta anche la classe *SMSServerConnectionImpl*, ma non ne abbiamo bisogno perché essa viene utilizzata per la ricezione degli SMS.
Visto che l'argomento può risultare ostico e i lettori

potrebbero avere difficoltà a riprodurre i file sorgenti delle due classi riscritte, vi invitiamo a consultare il codice allegato all'articolo per avere una visione più completa.
Cominciamo dalla classe *Protocol*:

```
package com.symbian.midp.io.protocol.vsms;
import com.symbian.gcf.*;
import com.symbian.javax.wireless.messaging.
                                    SMSConnectionSession;
import java.io.IOException;
import javax.microedition.io.Connection;
public final class Protocol extends ProtocolBase {
  public Protocol() {
    super("sms", "vsms");
  }
  public void ensurePermission(String aName) {
    // qui non facciamo niente}
  public Connection openConnection(URI aUri,
                           int aMode, boolean aTimeouts)
  throws IOException {
    if(aUri.toString().startsWith("vsms"))
      aUri = new URI(aUri.toString().substring(1));
      com.symbian.gcf.ConnectionSession session =
                    SMSConnectionSession.getSession();
      String host = aUri.getHost();
      ConnectionEndPoint connection;
    if(isServerConnection(aUri))
      connection = null;
    else {
      if(aMode == 1)
        throw new IllegalArgumentException();
        connection = new SMSClientConnectionImpl(
                                    session, aUri, 2);
     }
     connection.open();
     return connection;
  }
  protected boolean isServerConnection(URI aUri) {
    return aUri.getHost().length() == 0;
  }
}
```

In questa classe il metodo *ensurePermission(String aname)* lo implementiamo come un metodo vuoto. Per il resto questa classe non fa altro che stabilire una connessione invocando *SMSClientConnectionImpl*:

```
package com.symbian.midp.io.protocol.vsms;
import com.symbian.gcf.*;
import com.symbian.javax.wireless.messaging.*;
import com.symbian.util.NativeError;
import java.io.IOException;
import java.io.InterruptedIOException;
import javax.wireless.messaging.*;
class SMSClientConnectionImpl extends
                          DatagramConnectionEndPoint
implements MessageConnection {
[...]
        protected static void checkSecurity(String
          aPermission, String aPermissionArgs[]) {
        }
}
```

Come si può vedere questa è la classe per la creazione di un nuovo messaggio e per l'invio. Nel codice è stata lasciata tutta la parte relativa ai permessi tranne poi rendere vuoto il contenuto del metodo *checkSecurity*, che è quello che genera la fastidiosa richiesta di conferma all'utente.
Ora possiamo ricompilare le nostre classi e creare il nuovo jar. Non bisogna però installarlo sul telefono. Infatti dopo aver installato la versione senza hacking bisogna sostituire il jar originale con quello modificato. Per localizzare la midlet sul telefono (sarà in una posizione del tipo "E:/System/MIDlets/[12345678]") per la Series60 si può utilizzare FExplorer.
Fatto questo siamo pronti a lanciare l'applicazione e ad inviare messaggi senza che ce ne venga chiesta conferma. La procedura può sembrare poco ortodossa ma funziona!

INVIO SCHEDULATO DEGLI MMS

La MIDlet che utilizziamo per inviare MMS è *MMSMIDlet*. Progettiamo questa MIDlet in modo che all'avvio dell'applicazione compaia un *Form* contenente due *TextField*, uno per il subject del messaggio e l'altro per il numero di telefono o l'indizzo email del destinatario, uno *StringItem* che riporta il numero di allegati del messaggio e, come per gli sms, un *DateField* dove viene visualizzata l'ora e la data attuale. Se l'utente non modifica tale impostazione l'mms sarà spedito immediatamente, altrimenti sarà inviato alla data e ora di spedizione. Vediamo come costruire questi campi e come inserirli nel *Form*:

```
Form mmsForm = new Form("MMS");
TextField subjectField = new TextField("Subject:", null,
                                   256, TextField.ANY);
TextField destinationField = new TextField("Destinatario:
                               ", null, 256, TextField.ANY);
StringItem partsLabel = new StringItem("Allegati:", "0");
DateField sendingDateField = new DateField("Ora e data
             ora di spedizione:", DateField.DATE_TIME);
Date actualDate = new Date();
sendingDateField.setDate(actualDate);

mmsForm.append(subjectField);
mmsForm.append(destinationField);
mmsForm.append(partsLabel);
```

```
mmsForm.append(sendingDateField);
```

Una volta che l'utente ha inserito queste informazioni deve aggiungere gli allegati al messaggio. Il *Form* ha pertanto un *Command* ("Inserisci oggetto") che consente di visualizzare una lista delle tipologie di allegati: testo, immagine, audio, video. In particolare quando si invia il comando *"Inserisci oggetto"* viene chiamata la classe *PartsDialog* e il suo metodo *show( )*:

```
if (c = = CMD_ADD_PART) {
if (partsDialog = = null) {
            partsDialog = new PartsDialog(this);
            }
            partsDialog.show();
 }
```

La classe *PartsDialog* è quella che si occupa della gestione degli allegati e il metodo *show( )* mostra la lista delle tipologie di allegati. Si tratta di una lista a scelta esclusiva, per cui bisogna selezionare una delle tipologie:

```
public PartsDialog(MMSMIDlet mmsMIDlet) {

       this.mmsMIDlet = mmsMIDlet;
       String[] stringArray = {"Testo", "Immagine",
                                      "Audio", "Video"};
       typeList = new List("MMS", Choice.EXCLUSIVE,
                                      stringArray, null);
       typeList.addCommand(CMD_BACK);
       typeList.addCommand(CMD_NEXT);
       typeList.setCommandListener(this);

   }

public void show() {
       mmsMIDlet.getDisplay().setCurrent(typeList);
}
```

La scelta della tipologia di testo provoca la chiamata della classe *TextDialog* di *PartsDialog*, un *Form* contenente una *TextField* dove inserire il testo del messaggio. Una volta inserito il testo, i dati vengono incapsulati in un array di byte e codificati secondo la codifica *UTF-8*. A questo punto l'allegato sarà memorizzato in un oggetto *MessagePart*, fornito dalle WMA e il cui costruttore da noi utilizzato è:

```
MessagePart(byte[], int, int, String, String, String, String)
```

I parametri di questo costruttore sono rispettivamente un array di byte contenente i dati, l'offset (posizione di inizio dei dati all'interno dell'array), la lunghezza in byte dei dati, il *MIME content-type ("text/plain", "image/png", "audio/midi", "video/mpeg", ecc.)*, il *content-id* dell'allegato (es. *"id1"*, *"id2"*, ecc.), il *content-location* dell'allegato (*"null"* per il testo, il path del file per gli altri allegati), il tipo di codifica dell'allegato.

L'oggetto *MessagePart* può generare un'eccezione *SizeExceededException* nel caso in cui la dimensione dell'allegato superi quella consentita.

Costruito l'allegato viene visualizzato il *Form* iniziale in cui il numero di allegati viene aggiornato:

```
mmsMIDlet.getMessage( ).addPart(new MessagePart(
                contents, 0, contents.length,mimeType,
                        "id" + counter, null, encoding));
counter++;
mmsMIDlet.show( );

void show() {

partsLabel.setText(Integer.toString(
                                  partsDialog.counter));
   display.setCurrent(mmsForm);
}
```

Supponiamo ora di voler allegare un file, cioè un'immagine, un audio o un video.

Ripetiamo la procedura precedente con scelta dalla lista e stavolta è invocata la classe *ImageDialog* o *AudioDialog* o *VideoDialog* di *PartsDialog*.

Per poter selezionare un file all'interno del telefono facciamo ricorso alle *FileConnection API*, ovvero una serie di funzionalità opzionali che J2ME mette a disposizione per poter accedere al file system.

L'idea è quella di utilizzare queste API per costruire un file browser che ci permetta di scorrere all'interno della memoria del telefono e di un'eventuale memory card fino a selezionare il file desiderato. Costruiamo pertanto una classe thread di appoggio, FileBrowser che viene chiamata all'interno del costruttore di *ImageDialog* o *AudioDialog* o *VideoDialog*:

```
FileBrowser fb = new FileBrowser(this, mmsMIDlet,
                                          mimeType);
fb.start();
```

dove la stringa *mimeType* vale rispettivamente "image/*", "audio/*", "video/*". Il metodo *run ( )* di *FileBrowser* invoca *showCurrDir( )*. Questo metodo si occupa proprio di mostrare il contenuto della directory corrente e applicandolo ricorsivamente avremo il nostro browser.

Vediamo allora come opera questo metodo:

```
private void showCurrDir() {
   Enumeration e;
   List browser;
   try {
```

```
    if (MEGA_ROOT.equals(currDirName)) {
        e = FileSystemRegistry.listRoots();
        browser = new List(currDirName,
                                List.IMPLICIT);
    } else {
        currDir = (FileConnection)Connector.open(
                "file://localhost/" + currDirName,
                                Connector.READ);
        e = currDir.list();
        browser = new List(currDirName,
                                List.IMPLICIT);
        browser.append(UP_DIRECTORY, dirIcon);
        browser.addCommand(back);
    }

    while (e.hasMoreElements()) {
        String fileName = (String)e.nextElement();
        if (fileName.charAt(fileName.length()-1)
                                == SEP) {
        // questa è una directory
        browser.append(fileName, dirIcon);
    } else {
        // questo è un file
        browser.append(fileName, fileIcon);
    }
  }

  browser.setSelectCommand(open_select);
  browser.addCommand(exit_browser);
  browser.setCommandListener(this);
  if (currDir != null) {
    currDir.close();
  }
  msMIDlet.getDisplay().setCurrent(browser);
} catch (IOException ioe) {
    ioe.printStackTrace();
  }
}
```

Inizialmente, cioè al livello più alto del file system, il metodo *FileSystemRegistry.listRoots( )* fornito dalle FC API restituisce un oggetto *Enumeration* contenente la lista delle root presenti sul file system. Questi oggetti vengono aggiunti ad un List per poter riprodurre visivamente la lista delle root sul telefono.
Una volta selezionata una root il *CommandAction* invoca il metodo che serve ad "attraversare" le directory: *traverseDirectory(String)*. Questo metodo riconosce se l'operazione richiesta è verso l'alto (uscire dalla directory corrente) o verso il basso (entrare nella directory corrente), quindi chiama *showCurrDir( )* per la visualizzazione del contenuto della directory.
Questa volta *showCurrDir ( )* costruisce un oggetto *FileConnection* attraverso il metodo *Connector.open*, che ha come argomenti una url nel formato *file://<host>/<path>* (ad esempio *file://C:/Nokia/Images/image001.jpg*) e la modalità di apertura del file (lettura, scrittura o entrambi).
Anche in questo caso potete provare a realizzare un hacking per evitare che ogni volta che entrate in una directory o aprite un file vi venga chiesto il permesso!
Dall'oggetto *FileConnection* tramite il metodo *list( )* si ottiene nuovamente un *Enumeration* e si procede come prima. Tutto ciò viene ripetuto fino a quando l'oggetto selezionato non è una directory ma un file (ciò può essere facilmente riconosciuto dal fatto che il path di una directory termina con il separatore "/"). È nel *CommandAction* che viene effettuato questo riconoscimento e, se in effetti l'operazione eseguita dall'utente non è del tipo *"Apri directory"* ma del tipo *"Seleziona file"*, viene invocato il metodo *selectFile(String currFile)*:

```
private void selectFile(String fileName) {
  try {
    String currContentLocation =
        "file://localhost/" + currDirName + fileName;
    FileConnection fc = (FileConnection)Connector.
        open(currContentLocation,Connector.READ);
    if (!fc.exists()) {
      throw new IOException("Il file non esiste");
    }
    InputStream is = fc.openInputStream();
    fc.close();
    this.setContent(is);
    this.setContentLocation(currContentLocation);
    if (mimeType.startsWith("image")) {
        PartsDialog.ImageDialog imageDialog =
                    (PartsDialog.ImageDialog)dialog;
        imageDialog.play();
        mmsMIDlet.getDisplay().
                        setCurrent(imageDialog);
      } else if (mimeType.startsWith("audio")) {
        PartsDialog.AudioDialog audioDialog = (
                    PartsDialog.AudioDialog)dialog;
        audioDialog.play();
        mmsMIDlet.getDisplay().
                        setCurrent(audioDialog);
        is.close();
      } else if (mimeType.startsWith("video")) {
        PartsDialog.VideoDialog videoDialog =
                    (PartsDialog.VideoDialog)dialog;
        videoDialog.play();
        mmsMIDlet.getDisplay().
                        setCurrent(videoDialog);
        is.close();
      }
      ...
```

Come si vede viene costruito un *FileConnection* sul file selezionato, in modo da aprirlo in lettura, e poi il contenuto del file viene inserito in un *InputStream* utilizzando il metodo *openInputStream( )*

di *FileConnection*. Le FileConnection API permettono anche una gestione smart del file system che possiamo applicare perfettamente al nostro caso. Possiamo pensare, infatti, invece di dover scorrere tutto il file system per selezionare un file, di circoscrivere la ricerca ad esempio per le immagini alla directory delle immagini, ai video alla directory dei video, ai file audio alla directory degli audio. Infatti, utilizzando delle proprietà del sistema si possono ricavare le URL di particolare directory:

- **fileconn.dir.photos:** directory dove vengono salvate le foto o le immagini
- **fileconn.dir.videos:** directory dove vengono salvati i video registrati o scaricati
- **fileconn.dir.audios:** directory dove vengono salvati audio (comprese suonerie) registrati o scaricati:

Le URL ricavate possono essere poi passare al metodo *Connector.open*:

```
String imagesDir =
        System.getProperty("fileconn.dir.photos");
FileConnection fc = (FileConnection)Connector.
        open(imagesDir,Connector.READ);

String videoDir = System.getProperty
        ("fileconn.dir.videos");
FileConnection fc = (FileConnection)Connector.
        open(videoDir,Connector.READ);

String audioDir = System.getProperty
        ("fileconn.dir.tones");
FileConnection fc = (FileConnection)Connector.
        open(audioDir,Connector.READ);
```

Questo è quindi un metodo alternativo alla nostra implementazione, che per completezza didattica realizza un file browser vero e proprio.
A questo punto possiamo pensare di visualizzare l'immagine, ascoltare l'audio o riprodurre il video prima di allegarlo al messaggio. Quindi in base al content-type del file viene chiamato il metodo *play()* di *ImageDialog*, *AudioDialog* oppure *VideoDialog*. Cominciamo con *ImageDialog*, il metodo *play()* apre l'immagine e la visualizza:

```
InputStream content = fb.getContent();
Image img = null;
try
{
   img = Image.createImage(content);
   content.close();
}
catch (IOException e)
{ e.printStackTrace();}
this.append(img);
```

Ora basterà premere *"OK"* dal proprio telefono e come nel caso del testo verrà costruito l'oggetto *MessagePart*, per il quale a differenza di prima l'allegato è non un array di byte ma un *InputStream*. Lo stesso vale per gli allegati audio e video.
Diamo ora uno sguardo a come viene implementata la riproduzione degli allegati audio e video. L'oggetto utilizzato per la riproduzione è il *Player*. Il MIDP 2.0 contiene l'*Audio Building Block* che supporta la riproduzione di toni, sequenze di toni e file wav. Per audio diversi (midi, mp3, au) e video, abbiamo bisogno di un telefono che supporti le MMAPI *(Mobile Media API)*. Con queste API possiamo realizzare dei player completi, dotati di controllo di volume, velocità, ecc.
In questo caso, poiché il nostro scopo è un altro, ci limiteremo ad utilizzare il *Player* per una semplice riproduzione audio o video.
Partiamo dall'audio:

```
InputStream content = fb.getContent();
String contentLocation = fb.getContentLocation();
try {
   Player player = Manager.createPlayer(
                              contentLocation);
   player.start();
} catch(IOException ioe) {
     ioe.printStackTrace();
   } catch(MediaException me) {
     me.printStackTrace();
   }
```

Basta chiamare il metodo *start()* del *Player* per riprodurre l'audio. Per la riproduzione video utilizziamo un *VideoControl* per accedere al display del telefono:

```
InputStream content = fb.getContent();
String contentLocation = fb.getContentLocation();
try {
   Player player = Manager.createPlayer(
                              contentLocation);
   player.realize();
   VideoControl vc = (VideoControl)(
              player.getControl("VideoControl"));
   Item videoItem = (Item) vc.initDisplayMode(
         GUIControl.USE_GUI_PRIMITIVE, null);
    append(videoItem);
    player.start();
    vc.setVisible(true);
    content.close();
} catch(IOException ioe) {
    ioe.printStackTrace();
} catch(MediaException me) {
    me.printStackTrace();
}
```

Supponendo ora di aver inserito tutti gli allegati

che desideravamo, l'applicazione ci riporta al *Form* iniziale dove viene visualizzato il subject, il destinatario e il numero di allegati e non ci resta che dare il comando "Invia". Quando facciamo ciò viene chiamato il metodo *preSending ()* di *MMSMIDlet* che prepara il messaggio per poi passarlo alla classe *MMSsending*:

```
private void preSending() {
  try {
    String address = "mms://" +
                          destinationField.getString();
    message.setSubject(subjectField.getString());
    message.setDestination(address);
    Date sendingDate = sendingDateField.getDate();
    String statusMessage = "Invio messaggio a
                              " + address + "...";
    sendingMessageAlert.setString(statusMessage);
    Timer sendingTimer = new Timer();
    TimerTask sendingTask = new MMSsending(
                          message, sendingDate);
    sendingTimer.schedule
          (sendingTask,sendingDate); sleepApp();
  }
  catch (IllegalArgumentException iae) {
     errorMessageAlert.setString(iae.getMessage());
     display.setCurrent(errorMessageAlert);
  }
```

L'oggetto *message* è di tipo *MMSMessage*, che è una classe di appoggio contenente i metodi ausiliari per settare o restituire subject, destinatario e allegati del messaggio. Questo oggetto è quello che poi viene preso in consegna da *MMSsending* per l'invio vero e proprio.
All'interno del metodo *preSending ()* costruiamo un *Timer* ed un *TimerTask*, ovvero l'azione da eseguire quando il *Timer* scade, e scheduliamo il task alla data e ora decisa dall'utente.
Il *TimerTask* è proprio *MMSsending*, perciò allo scadere del *Timer* viene eseguito il metodo *run () di MMSsending*:

```
public MMSsending(MMSMessage message, Date
                              sendingDate) {
  this.message = message;
  this.sendingDate = sendingDate;
}

public void run() {
  String address = message.getDestination();
  MessageConnection mmsconn = null;
  try {
    mmsconn = (MessageConnection)
                     Connector.open(address);
    MultipartMessage mmmessage =
                        (MultipartMessage)
    mmsconn.newMessage(MessageConnection.
                         MULTIPART_MESSAGE);
    mmmessage.setAddress(address);
    MessagePart[] parts = message.getParts();
    for (int i = 0; i < parts.length; i++) {
      mmmessage.addMessagePart(parts[i]);
    }

     mmmessage.setSubject(message.getSubject());
        mmsconn.send(mmmessage);
     } catch (Exception e) {
        e.printStackTrace();
     }

      if (mmsconn != null) {
        try {
           mmsconn.close();
        }
        catch (IOException ioe) {
           ioe.printStackTrace();
        }
      }
   }
```

Come per gli sms anche l'invio di mms è implementato da un'interfaccia *MessageConnection*. Per aprire la connessione l'applicazione ottiene un oggetto che implementa il *MessageConnection* dalla classe *Connector*, attraverso una URL che identifica l'indirizzo.
Per quanto riguarda il messaggio vero e proprio, esso è rappresentato dall'interfaccia *MultipartMessage*. Si costruisce quindi un nuovo *MultipartMessage* a partire dal *MessageConnection* e poi si aggiungono i vari allegati attraverso il metodo *addMessagePart*. A questo punto utilizzando il metodo send di *MessageConnection* il messaggio viene spedito.

CONCLUSIONI

In questo articolo abbiamo visto come, utilizzando le WMA, sia possibile inviare sms e mms su J2ME. Inoltre abbiamo reso più interessante l'applicazione realizzata fornendo la possibilià di fare degli invii programmati.
Abbiamo inoltre analizzato l'hacking del protocollo *"sms"* al fine di evitare la richiesta all'utente di eseguire classi non sicure.
Infine abbiamo mostrato come utilizzare le MMAPI per riprodurre file audio e video e le FileConnection API per accedere al file system.
D'ora in poi se una mattina avete deciso di dormire invece di andare a lavoro, anziché svegliarvi per mandare un sms al vostro capo scrivendogli "Sto male, oggi resto a casa", preparate il messaggio la sera prima e la mattina dormite pure!

Vincenzo Viola

SOFTWARE SUL CD

ACE
LA LIBRERIA MULTIPIATTAFORMA PER LA RETE

Di Ace ce ne parla in modo approfondito Alfredo Marroccelli nel bell'articolo contenuto in questo stesso numero di ioProgrammo. Si tratta di una libreria che fornisce al C++ uno strato d'astrazione verso il TCP/IP. In parole povere consente di programmare applicazioni che fanno un uso estensivo delle risorse di rete, costruendo uno strato superiore che fa da interfaccia verso il sistema operativo. Questo garantisce la massima portabilità da un sistema all'altro. Di fatto in fase di programmazione sarà semplicemente necessario richiamare le funzioni e i metodi esposti da ACE, sarà poi ACE ad occuparsi di capire quale sistema c'è sotto.

Directory:/ Ace

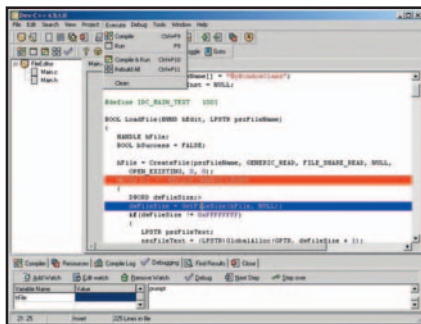
ECLIPSE 3.1.2
LA PIATTAFORMA UNIVERSALE MULTIFUNZIONE

Sembra un sottotitolo esagerato ed eclatante: *"La piattaforma universale multifunzione"* ed invece Eclipse è nato proprio con questo scopo.

A prima vista sembra un normale editor per programmatori Java, anzi molto di più che un normale editor, visto che ospita una serie di funzionalità piuttosto avanzate che vanno dal *code completion* alla *syntax highlithing* al *refactoring* e che lo stanno portando a diventare uno standard proprio per Java, tuttavia è anche vero che Eclipse è completamente estensibile per mezzo di plugin, tanto che può essere utilizzato da programmatori PHP come da programmatori C++ e persino come frontend verso database etc.
Insomma qualunque tipo di necessità voi abbiate, Eclipse è in grado di aiutarvi. Se poi siete dei programmatori Java rientra in quel ristretto numero di software indispensabili per gestire rapidamente il vostro lavoro.

Directory:/ Eclipse

DEV CPP 4.9.9.2
LA VERSIONE PIÙ RECENTE DELL'AMBIENTE PIÙ USATO DAI PROGRAMMATORI C++

Dev C++ è pratico, non ha costi, è molto funzionale leggero e versatile, questo lo rende un tool estremamente amato dai programmatori C++ e che si contrappone nella sua semplicità a strumenti ben più costosi.

È ovviamente dotato di tutte le funzionalità tipiche di un ambiente di programmazione come il *sintax highliting* e il *code complexion*, ma soprattutto è un'indispensabile se ritenete di volere o dover programmare in C++.

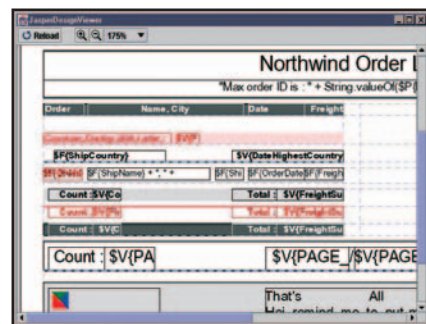
Directory:/ DevCPP

JDK 1.5.0 UPDATE 6 CON NETBEANS
LO STRUMENTO INDISPENSABILE PER SVILUPPARE IN JAVA

Se siete sviluppatori Java o avete intenzione di imparare a programmare in Java avete sicuramente bisogno del JDK. La versione che vi proponiamo è particolarmente interessante, perché oltre al consueto JDK contiene un bundled una versione di NetBeans, ovvero un ambiente di programmazione appositamente studiato per applicazioni Java. NetBeans è piuttosto potente, anche se molto pesante. È necessario avere un computer performante per lavorare in tranquillità. E d'altra parte il concorrente "Eclipse" non è certo un mostro di leggerezza. In ogni caso le caratteristiche di Eclipse sono interessanti. Si va dal code completion alla sintassi highliting al refactoring.
Sicuramente è un ottimo editor alternativo ad Eclipse ed altamente estendibile grazie ai moduli.

Directory:/ J2SE

JASPER REPORTS 1.2.0
UNA LIBRERIA SCRITTA IN JAVA PER CREARE REPORT PERSONALIZZATI

Un libreria per generare report in grado di inviare il proprio output allo

schermo, alla stampante o addirittura a file in formato PDF, HTML, XLS, CSV e e XML.
Interamente scritta in Java, può facilmente essere utilizzata in un infinità di applicazioni, per generare contenuti dinamici.
Directory:/ JasperReports

MRTG 2.13
MISURAZIONI DI RETE PERFETTE

Volete sapere quanto consuma in termini di banda il vostro WebServer? Avete necessità di conoscere quanta banda viene consumata in termini di accessi FTP? Mrtg è un frontend verso il servizio SMTP. Consente di misurare con estrema precisione tutti i parametri che caratterizzano il vostro sistema sia esso un sistema Window che Linux. Requisiti fondamentali per utilizzare a fondo questo frontend sono un'installazione di Perl funzionante sulla propria macchina e che il servizio Snmp sia attivo.
Directory:/ Mrtg

MYSQL 5.0.18
UNO DEI DB SERVER PIÙ USATI AL MONDO

Se Apache e PHP fanno da supporto alla maggior parte delle applicazioni che girano oggi su Internet, è anche vero che quasi tutte queste applicazioni sfruttano almeno in parte un database MySQL come sostegno per conservare i dati che manipolano. MySQL è estremamente leggero, veloce, flessibile, inoltre è multipiattaforma. Offre caratteristiche di tutto rispetto, che vanno dal supporto alle transazione alla ricerca full text, è perfettamente integrato con PHP. Tutte queste ragioni ne hanno fatto un leader per quanto riguarda il settore dei database.

In questo numero oltre alla versione 4.1.11 ormai consolidata vi presentiamo anche la nuovissima Beta 5.0.3 da non utilizzare in ambienti di produzione, ma piuttosto intrigante per capire da soli qual è la direzione che uno dei database più usati al mondo sta usando.
Directory:/ MySQL

PHP 5.1.2
IL LINGUAGGIO DI SCRIPTING PER IL WEB

Ormai PHP lo conoscete tutti, e se non lo avete mai usato, sicuramente vi sarà capitato di accedere a qualche sito web sviluppato con PHP. Saprete dunque perciò che è un linguaggio di scripting particolarmente utilizzato per sviluppare Web Application. Incredibilmente potente, fa della completezza del linguaggio, della facilità di apprendimento, della capacità di integrarsi con applicazioni di Dababase i suoi punti di forza.
Directory:/ PHP

WXWIDGETS 2.6.2
COMODE LIBRERIE PER LO SVILUPPO DI INTERFACCE GRAFICHE

Interessantissime queste librerie, più volte le abbiamo utilizzate all'interno dei nostri programmi per realizzare degli esempi. Si tratta di librerie che consentono la creazione di interfacce grafiche, possono essere utilizzate da C++ ma anche da altri linguaggi come ad esempio Python. La cosa estremamente interessante è che consentono lo sviluppo di applicazioni completamente multipiattaforma, sono disponibili infatti sia in ambiente *nix che in ambiente Windows. Continueremo sicuramente a parlarne in modo intensivo da queste pagine.
Directory:/ wxWidgets

PHPMYADMIN 2.8.0
MOLTO PROBABILMENTE IL FRONTEND PIÙ USATO AL MONDO PER MYSQL

Ci possono essere un milione di motivi per cui usare un'applicazione Web come frontend verso MySQL, ad esempio quando il vostro provider supporta solo la connessione a localhost è assolutamente indispensabile. PHPMyAdmin tuttavia non è un ripiego a un'interfaccia grafica evoluta, anzi la complessità delle funzioni che esporta lo rende molto spesso preferibile a un'interfaccia standalone.
Directory:/ phpmyadmin

OPENCV
LA LIBRERIA MULTIPIATTAFORMA PER LA WEBCAM

Ce ne parla a lungo Antonino Panella, nell'articolo che questo mese illustra come creare un programma di tiro al bersaglio. L'esempio è didattico, ma le OpenCV sono delle librerie opensource che consentono di gestire in modo ottimale la webcam. Le applicazioni sono infinite, si va dal riconoscimento facciale al motion detection e persino ad usi avanzati e futuristici come il riconoscimento del labiale. Si tratta di librerie estremamente potenti che consentono di lavorare con le immagini in modo sofisticato. Un primo esempio d'uso lo trovate in questo numero, ma siamo convinti che voi stessi sarete in grado di suggerirci impieghi molto fantasioso di questo ottimo framework
Directory: OpenCV

PRTG TRAFFIC GRAPHER 5.2.0.565
TRAFFICO SOTTO CONTROLLO

Mrtg lo conoscete tutti, si tratta di un software estremamente preciso per determinare il consumo di banda, di processore, di memoria di un sistema. MRTG interroga una macchina usando il protocollo SNMP e questo garantisce una resa ottimale . MRTG ha degli svantaggi. Si tratta di un'applicazione perl che non ha la stessa facilità di utilizzo di un'applicazione a finestre. Per gli utenti Windows c'è PRTG, che tramite una comoda GUI esporta tutte le funzionalità classiche di MRTG in modo comodo e intuitivo, con tanto di grafici e report dettagliati. Molto utile.
Directory: /Prtg

SORTING, ALGORITMI AVANZATI

L'ORDINAMENTO DEI DATI È UN'OPERAZIONE CHE RICORRE DI FREQUENTE. UN ALGORITMO EFFICIENTE PUÒ QUINDI SEGNARE LA DIFFERENZA TRA UNA BUONA E UNA CATTIVA APPLICAZIONE

Ordinare dati, che siano essi espressi come: array, liste di puntatori o come record di una tabella di database; è forse l'attività più diffusa nell'ambito della programmazione di computer. La vastissima letteratura a riguardo ne è una riprova. Certo è che l'argomento sorting non si può facilmente liquidare con la sterile enunciazione di uno tra tanti metodi negli anni approntati. Sempre più spesso capita di fare i conti con situazioni imprevedibili e difficilmente classificabili che richiedono l'applicazione personalizzata di algoritmi di ordinamento. Ad esempio, con i database non sempre è sufficiente cliccare il bottoncino presente sul DBMS di turno per ordinare i dati rispetto ad un attributo. Ne tanto meno si possono facilmente risolvere alcuni problemi con semplici query SQL. Per questo primo appuntamento ci occuperemo della sintetica e quasi manualistica descrizione dei metodi più efficienti nel ambito del sorting. Potranno essere un utile supporto per il programmatore. Nel prossimo appuntamento affronteremo specifici problemi legati ad argomenti come SQL, i puntatori oppure la ricerca binaria.
E' altrettanto vero che i linguaggi moderni dispongono di costrutti ad alto livello per l'ordinamento dei dati, tuttavia per lo scopo didattico di questa rubrica e per lo spirito di conoscenza che deve animare ogni programmatore, in questo numero ci divertiremo proprio a comprendere quali sono i vari algoritmi di ordinamento e quali siano i loro vantaggi e svantaggi rispetto alle applicazioni da sviluppare. Questi algoritmi sono estremamente datati, ma nel tempo nessuno ne ha trovato di migliori, segnale evidente di come nonostante la tecnologia si evolva continuamente rimangono alcuni pilastri di base che costituiscono anche quel bagaglio di conoscenze che ogni programmatore che voglia affrontare questo mondo con professionaità e in modo produttivo deve portare sempre con se

HEAP SORT

Consideriamo il vettore disordinato:

7	15	4	1	3

Per semplicità ad ogni passata la testa dell'insieme chiamato heap verrà ricopiata su di un secondo vettore. A rigore tale vettore non è indispensabile. Inizialmente bisogna costruire l'heap. Operazione sviluppata per fasi come mostrato nella tabella di seguito (ogni riga rappresenta nelle sue prime cinque caselle il vettore e nell'ultima la fase del processo generativo). Con il trattino si indica che al momento la casella non contiene valori. Il simbolo asterisco segnala che in quella fase il vettore parziale corrispondente non è un heap.

Vettore (heap) - tem					Fase
7	-	-	-	-	1
7	15	-	-	-	2
7	15	4	-	-	3 *
4	15	7	-	-	4
4	15	7	1	-	5 *
4	1	7	15	-	6 *
1	4	7	15	-	7
1	4	7	15	3	8 *
1	3	7	15	4	9

Nella fase 1 viene ricopiato nell'heap il primo elemento del vettore. Ovviamente, questo insieme degenere (poiché costituito da un singolo numero) è un heap, infatti, rispetta le proprietà. Successivamente, si aggiunge il secondo numero. Anche il nuovo mini vettore è un heap, come si può notare *temp[2]>temp[1]*. Al terzo passaggio, l'aggiunta del nuovo numero rende il vettore in fase di costruzione non un heap, nella tabella l'evento è segnalato con asterisco. Bisogna quindi scambiare, ottenendo così il risultato di fase 4 che è un heap. Lo scambio non può produrre situazioni per le quali altri elementi dell'insieme non rispettino con la nuo-

va configurazione le proprietà dell'heap, poiché dopo gli scambi le disuguaglianze sono maggiormente verificate. L'aggiunta del quarto elemento impone un doppio scambio fornendo il doppio risultato prima di fase sei e poi di fase sette. In queste ultime fasi il vettore non costituiva heap poiché nel primo caso si presentava *temp[4]<temp[2]* e nel secondo caso *temp[2]<temp[1]*. L'ultimo inserimento, del quinto numero non genera, il ricercato mucchio poiché non è rispettata la condizione *temp[5]>temp[2]* quindi si procede allo scambio che sancisce l'ultimo passo per la produzione del vettore heap (fase 9). Terminata la fase di costruzione bisogna procedere con l'estrazione degli elementi dall'heap per la generazione del vettore ordinato. Ogniqualvolta si estrae un elemento dal mucchio (il primo perché il più piccolo parziale) bisogna ricomporre l'heap al vettore rimanente.

Vettore	1°	2°	3°	4°	5°	Fase
arr	1	-	-	-	-	1
temp	-	3	7	15	4	2 *
temp	4	3	7	15	-	3 *
temp	3	4	7	15	-	4
arr	1	3	-	-	-	5
temp	-	4	7	15	-	6
Temp	15	4	7	-	-	7 *
temp	4	15	7	-	-	8
arr	1	3	4	-	-	9
temp	-	15	7	-	-	10
temp	7	15	-	-	-	11
Arr	1	3	4	7	-	12
temp	-	15	-	-	-	13
temp	15	-	-	-	-	14
arr	1	3	4	7	15	15

Strutturiamo l'intero algoritmo:

```
Costruzione heap
Ripeti
        Estrazione dell'elemento di testa
                                dall'heap
        Ricostruzione heap
Finchè l'heap è vuoto
```

Con riferimento all'esempio esaminiamo il ciclo di ripeti. In questo nuovo prospetto oltre al vettore di nome *temp* che contiene l'heap è presentato il target ossia il vettore *arr* che man mano viene costruito è al termine risulterà totalmente ordinato.
Dopo aver riportato in arr il primo elemento (il più piccolo) bisogna ricomporre l'heap; si procede spostando l'ultimo elemento in prima posizione (fase 3), e applicando nuovamente la regola. Questa volta però, si ragiona esaminando inizialmente il primo elemento, dovrà quindi risultare che temp[1]<temp[2] e che temp[1]<temp[3]. Ovviamente, se l'heap fosse stato più "popolato" si sarebbe reso necessario il controllo delle caselle di indice multiplo a quelle appena analizzate, ma nel caso in esame non esistono. Si procede trovando il minimo tra i due elementi di indice due e tre ed effettuando se necessario lo scambio, come si rileva nella fase 4. Il risultato parziale ottenuto per come costruito è un heap, quindi si estrae il primo elemento lo si colloca nel target e si itera il procedimento. Una possibile implementazione può essere quella di seguito proposta.

```
void heapsort(int *arr)
{  int i,j,k,z,q,swh,w,h,im;
   int *temp;
// array temporaneo
//contenente  l'heap
   bool is_heap;
   // Costruzione heap
   for (i=1; i<=n; i++)
   {  temp[i]=arr[i];
          z=i;
          is_heap=false;
          while ((z>1) && (! is_heap))
          {  q= z/2; // quoziente
        if (temp[z]<temp[q])
        {  swh=temp[z]; temp[z]=temp[q];
temp[q]=swh;
              }
              else is_heap=true;
          }
   };

// ricostruzione del vettore ordinato
 for (i=1; i<=n; i++)
 {  arr[i]=temp[1];
    k=n-i;
    temp[1]=temp[k+1];
    is_heap=false;
    z=1;
    while ((! is_heap) && (2*z <= k))
```

HEAP SORT

Si tratta di un metodo popolare. Si basa sul concetto di heap, letteralmente mucchio. Un heap è un insieme di elementi (a1, a2, …, an) parzialmente ordinati. L'ordinamento parziale è assicurato dalla seguente proprietà:

ai >= ai/2

per ogni i appartenente all'intervallo [1,n]
con n cardinalità dell'insieme. La frazione è intesa come quoziente intero
Realizzare un algoritmo che implementi tale metodo significa, sulla base di un insieme di numeri disordinati, effettuare una opportuna permutazione in modo che costituiscano un heap, estrarre il primo elemento dall'heap, considerato che sarà il valore più piccolo e ripetere il procedimento sul restante insieme. Ad ogni passata si estrarrà il valore più piccolo che permetterà così la costruzione finale del vettore ordinato.

```
  {  w=2*z; h=w+1;
          im=h;
          if (h > k) im=w;
          else if (temp[w] < temp[h]) im=w;
          if (temp[z]>temp[im])
          { swh=temp[z]; temp[z]=temp[im];
 temp[im]=swh;
           z=im; }
          else
           is_heap=true;
  }
 }
}
```

Nel valutare la complessità dell'algoritmo bisogna tener conto dei due cicli distinti per la creazione e ricomposizione dell'heap. I due frammenti di codice hanno stessa complessità. Ma quanto vale ognuna? Nel caso peggiore bisogna trasferire un numero dal fondo fino al primo elemento lungo indici di posto man mano dimezzato n, n/2, n/4 fino ad arrivare al primo. Questa serie, come gli amanti della matematica possono confermare, è un logaritmo. Essendo il ciclo ripetuto n volte, in definitiva la complessità di un singolo stadio è $O(n*log(n))$ ed in totale, quindi, $O(2*n*log(n))$ che può essere approssimata a $O(n*log(n))$.

MERGE SORT

Inizialmente il vettore di *n* elementi viene diviso in due vettori di *n/2* elementi; i due sottovettori ottenuti vengono ordinati separatamente riapplicando il metodo, e successivamente fusi (merge). Ordinare i sottovettori di lunghezza *n/2* significa dividerli in sottovettori di lunghezza *n/4* e applicare lo stesso procedimento esposto per i sottovettori padre. La chiamata ricorsiva al metodo termina quando si perviene a sottovettori di lunghezza unitaria. Nello sviluppare l'algoritmo e la conseguente codifica C++ si separano le due fasi di ordinamento e di fusione. Le due partizioni sono individuate dagli intervalli *[sx,m]* e *[m,dx]*, tali indici sono anche i parametri della funzione. L'attuazione della fusione avviene attraverso tre fasi identificate da altrettanti cicli di *while*. In particolare, il primo ciclo effettua il *merge* vero e proprio, mentre i successivi due, si occupano di gestire i residui a seconda se presenti nella partizione sinistra o in quella destra. Il vettore temporaneo *temp* ricostruisce il vettore ordinato come giustapposizione (fusione) dei due sottovettori. L'ultimo ciclo di *for* si occuperà di ricopiare il risultato, vettore temporaneo, nell' intervallo *[sx,dx]* del vettore originario *a*.

```
void merge (int sx, int m, int dx)

{     int temp[20], i, j, k;
i=sx;
          j=m+1;
          k=sx;
          // Fusione (merge) dei
          // due vettori ordinati
          while ((i<=m) && (j<=dx))
          {   if (a[i]<a[j])
                              temp[k]=a[i++];
                              else temp[k]=a[j++];
                              k++;
          };
          // Gestione dei residui
          // (se presenti) sinistri
          while (i<=m)
          {   temp[k]=a[i++];
                              k++;
          };
          // Gestione dei residui (se presenti)
                                             sinistri
          while (j<=dx)
          {          temp[k]=a[j++];
                              k++;
          };
          // Ricomposizione del vettore a partire da
quello temporaneo
          for(i=sx; i<=dx; i++)
          a[i]=temp[i];
}
```

La procedura ricorsiva mergesort prende come parametri due variabili sx e dx che indicano rispettivamente l'estremo sinistro e destro del vettore (o in generale della porzione di vettore) e ordina la sottosequenza compresa nel sottovettore delimitato da tali indici. L'ordinamento viene fatto dalla sequenza di tre operazioni: ordinamento della partizione sinistra (chiamata alla prima procedura mergesort), ordinamento della partizione destra (chiamata alla seconda procedura mergesort) e fusione dei due sotto-

MERGE SORT

La filosofia del divide et impera (frase storica "dividi per dominare") è realizzata appieno nel metodo di ordinamento conosciuto come merge sort. Si tratta di iterare il processo di partizionamento, ordinamento e fusione. Nella fase di ordinamento si innesca il carattere ricorsivo. Ecco come può essere schematizzato:

```
detsort (sequenza)
{   se sequenza (è disordinata)
oppure (più lunga di 1) allora
    { partiziona in due liste sinistra e
destra
        detsort(sinistra)
            detsort(destra)
            fondi(sinistra, destra)
    }
}
```

vettori (chiamata a merge, precedentemente descritta). La catena delle chiamate ricorsive a mergesort ad un certo punto termina e si esce da una di esse, così si ripercorre a ritroso la stessa catena, fino ad ottenere il risultato sperato di ordinamento. Si esce dalla procedura quando risulta falsa la condizione dell'if, ossia quando, l'indice sinistro non risulta minore di quello destro, il che significa che i due estremi si sono incontrati e quindi il sottovettore in esame è di lunghezza minima 1. Il valore med indica la media dei due estremi, si noti che in C++ essendo med un intero, il risultato della divisione è sempre un numero intero, secondo le regole di cast proprie del linguaggio, tale operazione produrrà quindi, il quoziente.

```
void mergesort(int sx, int dx)
{    int med;
    if (sx<dx)
    {   med=(sx+dx)/2;

         // Chiamate ricorsive
         // per le due partizioni
         // sinistra e destra
            mergesort(sx,med);
            mergesort(med+1,dx);
       // Fusione (merge) delle due
      //partizioni ordinate
            merge(sx,med,dx);
         }
};
```

15	12	36	9	29	7	18	30	10

È evidente come la procedura merge presenti complessità proporzionale a n, poiché si tratta di una scansione lineare del vettore. Tale procedura è richiamata log(n) volte. Per comprendere ciò si pensi che ogni volta si divide il vettore a metà, quindi si generano partizioni di lunghezza n/2, n/4, n/8 e così via, il cui numero è pari, appunto, a log(n). In definitiva, la complessità totale del metodo è O(n*log(n)). Una versione efficiente del programma andrebbe sviluppata in modo non ricorsivo.

Fasi\indici	1	2	3	4	5	6	7	8	9	Pivot
0	15	12	36	9	29	7	18	30	10	29
1	15	12	10	9	18	7	29	30	36	10 e 30
2	7	9	10	12	18	15	29	30	36	7 e 18
3	7	9	10	12	15	18	29	30	36	12
4	7	9	10	12	15	18	29	30	36	

"Fasi del quick sort. Per ogni fase sono specificati gli elementi di pivot. Il colore verde indica le parti del vettore parzialmente ordinate. Nell'ultima fase l'intero vettore è ordinato".

QUICK SORT

Supponiamo di avere un vettore di nove numeri Bisogna individuare in una prima fase un elemento mediano. In conformità con la letteratura sull'argomento chiameremo tale elemento pivot. Come per il basket è necessario stabilire un numero che si trovi nel mezzo. Si tratta del numero 29. Ora si esaminano gli elementi a sinistra e a destra del pivot, a partire dal primo per l'insieme uno e dall'ultimo per il secondo. Ci si ferma quando si incontra un elemento in disordine con il pivot. Nella scansione a sinistra quando si incontra il numero 36, nella scansione a destra già il primo elemento esaminato è oggetto di scambio poiché più piccolo del pivot. Così i due numeri 36 e 10 vengono scambiati. La fase non è conclusa. Riprendendo a scorrere la parte sinistra non si incontrano altri elementi più grandi del pivot se non il pivot stesso che risulta dal confronto non minore di se stesso (essendo appunto uguale) quindi soggetto a scambio. A destra invece, il numero 18 risulta più piccolo del pivot. I due elementi 29 e 18 si scambiano e la prima fase termina. Ogni fase si conclude quando gli indici che percorrono le due partizioni si incontrano. Nella figura 1 sono riportate tutte le fasi ed i rispettivi pivot. Le partizioni sono proposte di colore diverso. L'ordinamento parziale è rappresentato in verde.
Da qui in avanti, in generale, si registrerà una certa asimmetria. Come si nota nella fase 1 sono stati individuati due insiemi rispetto al pivot che 29 (verde). Nel primo insieme tutti i numeri sono più piccoli rispetto al pivot, per il secondo sono invece maggiori del pivot. L'asimmetria sta nel fatto che gli insiemi hanno adesso (nel caso più generale) grandezze differenti. A questo punto bisogna applicare lo stesso metodo alle due liste ottenute. Per la prima il pivot è 10 mentre per la seconda è 30. Così l'ordinamento si ottiene in stadi successivi. Si evidenzia la natura ricorsiva dell'algoritmo che presenta sempre le stesse istruzioni applicate a sotto liste sempre più piccole.
Ecco il codice C++ applicato alla struttura dati array.

```
void quicksort (int sx, int dx)
{   int i,j,pivot,comodo;
    i=sx;  j=dx;
    // Calcolo del pivot
```

```
    pivot=a[(sx+dx)/2];
    do
    {
            // Ricerca a sinistra
           // di valori più grandi del  pivot

       while (a[i]<pivot) i++;

       // Ricerca a destra di valori
      //più piccoli del pivot
       while (pivot<a[j]) j--;

       // Se necessario si scambia

       if (i<=j)
       {   comodo=a[j]; a[j]=a[i]; a[i]=comodo;
           i++; j--;
       }
    }
    while (i<=j);
    // Chiamata ricorsiva alle due partizioni

 if (sx<j) quicksort (sx,j);
    if (i<dx) quicksort (i,dx);
}
```

Il vettore da ordinare è a, gli indici che scorrono sulle due partizioni destra e sinistra sono rispettivamente i e j. Il pivot viene calcolato banalmente come elemento centrale della sequenza. I due parametri sx e dx sono rispettivamente l'estremo sinistro e destro delle due partizioni

Ogni iterazione del ciclo gestisce il partizionamento e lo scambio. Al termine di tale ciclo le due partizioni sono parzialmente ordinate. Da notare le successive chiamate ricorsive rispetto alle due partizioni prodotte. L'algoritmo è ancora più rapido di quello che risulterà dalla analisi di complessità se si considera che le variabili più usate, come pivot, i e j, vanno (o possono essere esplicitamente tenute) in registri veloci o memorie cache.

La complessità dell'algoritmo è la risultante dei contributi dovuti dalla fase di partizionamento e dal numero di scambi. La prima delle due fasi consta di n confronti considerato che bisogna scandire l'intero array. Per sapere invece quanti scambi vengono effettuati è necessario effettuare un'analisi probabilistica. La probabilità che si verifichi una condizione di scambio è (n-pos+1)/n con pos posizione del pivot. Quindi, il numero atteso di scambi è pari alla somma di tutte le probabilità fratto n; che a seguito di semplificazioni algebriche ed approssimazioni risulta essere n/6. Nel caso fortunato in cui il pivot produca due partizioni di eguale lunghezza, allora il numero di passi sarà log(n), poiché si fa riferimento a partizioni ad ogni passo pari alla metà delle precedenti. In questa situazione il numero di confronti è n*log(n) ed il numero di scambi (n/6)*log(n). Il caso migliore che prevede sempre di selezionare la mediana nel processo di partizionamento ha probabilità bassa pari a 1/n, ad ogni modo il caso medio non presenta un deterioramento evidente della complessità dell'algoritmo. Un altro elemento importante nella valutazione della complessità risiede nel miglioramento delle prestazioni man mano che n aumenta. Nel caso peggiore, che peraltro si presenta raramente, quando il pivot corrisponde sempre ad uno dei due estremi della partizione, l'algoritmo degenera le sue prestazioni ad una complessità proporzionale a n2 (quindi poco quick!). È fondamentale la scelta del pivot che nell'algoritmo corrisponde all'elemento di mezzo. Nulla ci vieta di scegliere un qualsiasi altro elemento come il primo o l'ultimo. Ad ogni modo, test statistici hanno evidenziato un migliore comportamento nella scelta dell'elemento centrale. Hoare proponeva di scegliere il valore casualmente o come valore mediano di un campione opportunamente ottenuto da un altro algoritmo. Così la complessità nelle situazioni favorevoli rimane pressoché invariata, si migliorano invece, sensibilmente le prestazioni rispetto ai casi peggiori, che infondo sono lo spauracchio dell'utilizzatore di tale metodo.

CONCLUSIONI

La prossima volta affronteremo metodi personalizzati di ordinamento per la risoluzione di problemi non standard. Vi aspetto.

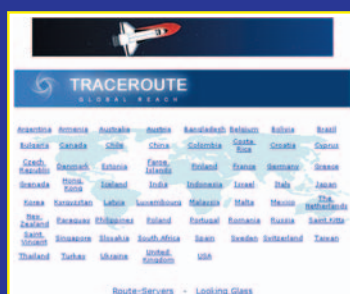
Fabio Grimali

QUICK SORT

Il nome è dovuto al suo creatore C.A.R. Hoare, che sviluppò il metodo come miglioramento (sostanziale!) del bubble sort. L'idea perseguita è di sviluppare un metodo che minimizzasse il numero di scambi a fronte di un aumento dell'efficienza. Il grande Nickaus Wirth, padre del linguaggio Pascal, e soprattutto padre della programmazione strutturata, nel suo "Algoritmi + strutture dati = programmi", individua come chiave dell'efficienza del quick sort il fatto che gli scambi tra i valori nel vettore avvengono in generale per lunghe distanze. Istruttivo è l'esempio nel quale si considera un vettore di n elementi sistemati esattamente nell'ordine inverso. Per ordinarli basterebbero solo n/2 scambi, ed esattamente quelli che coinvolgono il primo con l'ultimo elemento, il secondo con il penultimo e così via. È evidente che solo raramente il vettore si presenta in questa forma, ma l'esempio ha un valore simbolico visto che quick sort è una generalizzazione del concetto esposto.

ON LINE

DNSSTUFF

Per avere sempre sotto controllo il buon funzionamento del dns. E' possibile persino effettuare un check della corretta configurazione .http://www.dnsstuff.com/

TRACEROUTE

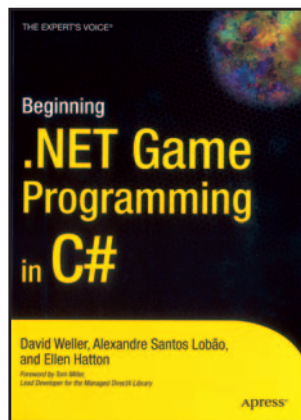
Per effettuare il corretto trace di un pacchetto TCP/IP. Consente di stabilire il percorso effettuato da un pacchetto prima che giunga a destinazione. http://www.traceroute.net

DEVLEAP

DevLeap è una community di professionisti dello sviluppo che vede la partecipazione dei maggiori esperti di programmazione italiana. La maggior parte dei componenti è infatti occupata in attività di formazione per alcune delle maggiori software house mondiali.

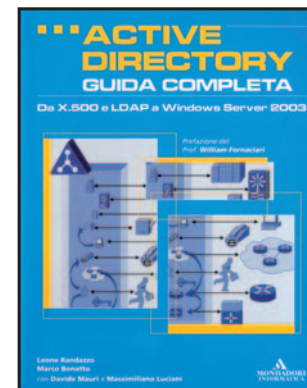
Biblioteca

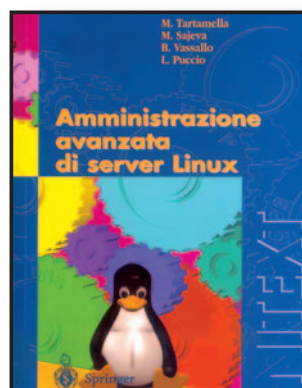
.NET GAME PROGRAMMING C#

In questo libro vengono illustrate le procedure per sfruttare al massimo il vostro compilatore .NET programmando i giochi con C#. In effetti le DirectX 9 sono direttamente accessibili da .NET allora perché non sfruttare le caratteristiche di un linguaggio ad alto livello per sviluppare videogiochi ? Il libro illustra lo sviluppo di cinque differenti progetti completi, ognuno dei quali risolve alcune problematiche essenziali e peculiari di un particolare tipo di videogames, in questo modo si fornisce un approccio molto scalabile che affronta in modo quasi indipendente la programmazione di videogiochi 3D e 2D. Il capitolo finale è decisamente innovativo, affronta infatti la scrittura di un videogioco per PocketPC. Gli esempi sono numerosi e ben curati così come è curato ogni aspetto della programmazione. In definitiva un buon libro, certamente utile per lo sviluppo di videogiochi ma non privo di interesse per chi vuole acquisire un punto di vista alternativo nell'utilizzo del C#.

Difficoltà: Alta • **Autore:** David Weller, Alexandre Santos Lobao, Hellen Hatton • **Editore:** Apress • **ISBN:** 1-59059-319-7 • **Anno di pubblicazione:** 2004 • **Lingua:** Inglese • **Pagine:** 414 • **Prezzo:** € 44,00

ACTIVE DIRECTORY GUIDA COMPLETA

Active Directory è una tecnologia che porta sulle spalle ormai almeno 4 anni di vita. Tuttavia non trova riscontro nelle lan delle piccole e medie imprese, viceversa è estremamente diffusa in lan di grandi dimensioni. Il torto di questa tecnologia è di essere complessa almeno quanto utile. In realtà utilizzare AD all'interno di ogni tipo di rete comporterebbe enormi vantaggi nella lan. Tutto risulta più omogeneo, dalla condivisione delle risorse alla pubblicazione delle informazioni alla gestione della sicurezza, rimane il tarlo di una certa difficoltà iniziale nella comprensione dei concetti di base.. Il libro è una guida certa, pratica e completa alla comprensione dei meccanismi della tecnologia Active Directory. Incentrato sullo studio di casi reali, lo rende particolarmente utile e pronto da essere utilizzato in contesti di produzione.

Difficoltà: Media • **Autore:** Leone Randazzo, Marco Bonatto • **Editore:** Mondadori Informatica • **ISBN:** 88-04-54023-0 • **Anno di pubblicazione:** 2005 • **Lingua:** Italiana • **Pagine:** 506 • **Prezzo:** € 40,00

AMMINISTRAZIONE AVANZATA DI SERVER LINUX

Linux rappresenta una fetta enorme dei sistemi server installati al mondo e come tale più di altri necessita di continui cambiamenti che orientino i suoi servizi verso un target che domanda servizi sempre di crescente qualit'. Questa crescente complessità della domanda, necessariamente provoca un innalzamento della complessità nella gestione del sistema. Questo libro affronta uno per uno i vari servizi possibili in ambito Linux. Non ha certo la pretesa di esaurire tutti i concetti legati a ciascun servizio, viceversa fornisce una base teorica che mette in grado l'utilizzatore di gestire un qualunque sistema linux indipendentemente dalla distribuzione o dal contesto in cui esso è installato. Non si tratta di un libro completamente pratico, tuttavia fornisce una base di conoscenza indispensabile per gestire con profitto ogni tipo di sistema in modo consapevole e senza abbandonarsi a tentativi che si rivelano il più delle volte un'arma a doppio taglio. Decisamente un'indispensabile

Difficoltà: Alta • **Autore:** M.Tartamella – M.Sajeva – B.Vassallo – L.Puccio • **Editore:** Springer • **ISBN:** 88-470-0234-6 • **Anno di pubblicazione:** 2004 • **Lingua:** Italiana • **Pagine:** 460 • **Prezzo:** € 35,00